MASTER EN SCIENCES INFORMATIQUES

Application des algorithmes génétiques au dilemme du prisonnier

Cooremans, Jérôme; Rondeaux, Nicolas

Award date:
1999

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'informatique
Année académique 1998-1999

Application des Algorithmes Génétiques au Dilemme du Prisonnier

Jérôme Cooremans
Nicolas Rondeaux

Mémoire présenté en vue de l'obtention
du grade de Licencié en Informatique

Remerciements

Tout au long de ce travail, nous avons eu la chance de bénéficier des conseils avisés de notre promoteur, Monsieur N. Habra. Nous tenons tout particulièrement à le remercier pour l'intérêt qu'il a porté à l'égard de notre travail.

Nous tenons également a remercier Monsieur J. P. Leclercq pour nous avoir aiguillé sur l'utilisation possible d'Algorithmes Génétiques dans notre travail.

Enfin, nous remercions également ceux qui nous ont soutenu et aidé à la réalisation de ce travail.

TABLE DES MATIERES

INTRODUCTION

L'objectif principal de ce mémoire était d'adapter la technique des algorithmes génétiques à la recherche rapide d'une ou plusieurs tactique(s) de jeu dans le cadre du dilemme du prisonnier itéré. Ceci, dans le but d'analyser leurs caractéristiques en relation avec leurs performances.

Pour ce faire, nous avons réalisé un programme permettant de générer de telles tactiques et de vérifier leurs performances. La structure des données et le fonctionnement du programme se basent sur nos données bibliographiques. Cette bibliographie traite principalement les deux sujets suivants : le dilemme du prisonnier, et les algorithmes génétiques qui sont respectivement exposés aux Chapitre I : p 5, Chapitre II : p 22.

Nous décrivons notre logiciel au Chapitre III : p 30 et présentons les expériences réalisées à l'aide de celui-ci, ainsi que les résultats obtenus au Chapitre IV : p 48.

L'avantage de notre logiciel est de permettre de générer et de tester rapidement des tactiques sans réfléchir à l'élaboration de celles-ci ; de plus ces tactiques ne sont pas prédéterminées et peuvent s'adapter au jeu de l'adversaire.

Chapitre I : LE DILEMME DU PRISONNIER

Dans cette synthèse bibliographique, nous présenterons le dilemme du prisonnier, ainsi que quelques variantes de celui-ci ; ensuite nous essayerons d'analyser diverses stratégies de jeu.

1.1. Introduction

Le terme « **dilemme du prisonnier** » provient de la situation suivante qui fut probablement imaginée : deux suspects armés ont été arrêtés devant une banque et incarcérés dans deux cellules différentes. Ils doivent très rapidement choisir entre avouer qu'ils allaient commettre un hold-up ou ne rien avouer. Les règles imposées par le juge sont les suivantes : si l'un avoue et l'autre pas, celui qui avoue sera libéré tandis que l'autre écopera de 5 ans de prison, si aucun n'avoue, ils seront tous deux condamnés à 2 ans de prison pour port d'arme illégal, si les deux avouent, ils iront chacun faire 4 ans de prison.

Il existe d'autres exemples où un tel dilemme peut apparaître ; comme dans le cas fictif où deux firmes décident de vendre le même produit sur un marché et où le prix de vente des produits et les ventes totales combinées des deux entreprises ne varient pas d'année en année ; ce qui varie c'est la part du marché que chaque firme s'approprie en fonction du budget qu'elle décide d'allouer à la publicité. Pour simplifier supposons qu'elles peuvent allouer 7 ou 9 millions de dollars ; si elles décident toutes les deux d'allouer 7 millions de dollars, elles feront chacune un bénéfice de 3 millions de dollars, si elles allouent toutes deux 9 millions de dollars à la publicité, elles recevront chacune un bénéfice de 1 million de dollars, car le marché est le même d'année en année et qu'elles perdent chacune 2 millions de dollars inutilement dans la publicité. Par contre, si une firme décide de n'allouer que 7 millions de dollars à la publicité alors que l'autre en alloue 9 millions de dollars ; cette dernière gagnera plus de parts de marché et fera de plus importants bénéfices, par exemple ; 4 millions de dollars et l'autre ne gagnera plus rien. Dans cette situation, on peut considérer que l'allocation par une firme d'un budget de 7 millions de dollars à la

publicité équivaut à coopérer (c) avec l'autre firme, tandis que l'allocation d'un budget de 9 millions de dollars équivaut à trahir (t) l'autre firme.

		Firme 2 **dépense** **7 millions$ (c)**	**dépense** **9 millions$ (t)**
Firme 1	**dépense** **7 millions$ (c)**	(3 millions$, 3 millions$) (c, c)	(0 millions$, 4 millions$) (c, t)
	dépense **9 millions$ (t)**	(4 millions$, 0 millions$) (t, c)	(1 millions$, 1 millions$) (t, t)

Tableau 1 : table des gains en fonction du choix des participants de coopérer (c) ou de trahir (t) pour un dilemme du prisonnier.

Pour qu'il y ait dilemme du prisonnier, il faut que : le bénéfice d'un traître non trahi (tc) > bénéfice de la coopération mutuelle (cc) > bénéfice de la trahison mutuelle (tt) > bénéfice du dupé (ct). L'exemple ci-dessus avec les deux firmes respecte bien ces inégalités (Tableau 1).

Le dilemme du prisonnier peut aussi être utilisé pour représenter d'autres situations (commerciales, sociales, politiques, etc.) où les acteurs se trouvent en face de deux choix similaires à une trahison et une coopération, comme par exemple la création des standards techniques, la course à l'armement, ...

En pratique, les gains possibles par chacune des parties sont rarement limités comme dans le dilemme du prisonnier ; en effet, rien n'empêcherait une firme d'allouer 20 millions de dollars pour la publicité ; dans ce cas elle perdra 7 millions de dollars puisque les parts de marché sont fixes. Le dilemme du prisonnier est donc le plus souvent une simplification des situations réelles ; en effet, on considère que les joueurs n'ont le choix qu'entre deux actions possibles : coopérer (c) ou trahir (t) et que les gains ou les pertes sont fixés pour chacune des 4 combinaisons possibles de jeux ((c, c) ; (c, t) ; (t, c) ; (t, t)).

Le dilemme du prisonnier tel que présenté ci-dessus ne tient pas compte de la possibilité qu'ont les participants de jouer plusieurs fois. C'est pourquoi d'autres variantes ont été développées. Dans le cas où les participants au dilemme du prisonnier peuvent jouer plusieurs parts, on parle de « **dilemme itéré du prisonnier** »; cependant, afin qu'il ne soit pas plus intéressant à deux entités de s'entendre pour, à

tour de rôle, trahir et se faire duper, (J.P.Delahaye, P.Mathieu, 10/06/1992) il faut rajouter la contrainte suivante : (tc+ct)/2<cc. Certains auteurs (J. P. Delahaye & Ph. Mathieu, 1993) ont introduit une variante du « **dilemme itéré du prisonnier** » pour laquelle, en plus de coopérer ou trahir, les deux joueurs ont la possibilité de renoncer (rr) à jouer. Ces variantes seront expliquées plus en détail par après.

Bien d'autres situations réelles peuvent être assimilées à la situation du « **dilemme itéré des prisonniers** ». Cependant, il peut être difficile d'évaluer les gains ou les pertes obtenus par les différentes parts pour les quatre combinaisons de jeux ((c, c) ; (c, t) ; (t, c) ; (t, t)). De plus, on peut imaginer d'autres options comme la renonciation, des degrés divers de trahison et de coopération, mais jusqu'à présent rien de très clair n'a pu être retiré de ces jeux à plus de deux options.

1.2. Différentes variantes du dilemme du prisonnier

Dans cette section, nous allons détailler les différentes variantes du dilemme du prisonnier, que sont le dilemme simple, le dilemme itéré.

1.2.1. Le dilemme simple

Dans le « **dilemme du prisonnier simple** » les participants ne jouent qu'une fois l'un contre l'autre, ce dilemme peut être modélisé de la façon suivante :

- Si les deux agents coopèrent, ils obtiennent un gain de récompense (cc).
- Si tous les deux trahissent, ils perçoivent une rémunération de punition (tt).
- Si l'un coopère alors que l'autre trahit, celui qui coopère obtient un gain de dupe (ct), alors que le second obtient un gain de trahison (tc).

Les quatre combinaisons de jeux sont reprises ci-dessous dans la matrice des payoffs (bénéfices) ; cette matrice devant être connue des deux participants avant le jeu.

	Coopération	Trahison
Coopération	(cc, cc)	(ct, tc)
Trahison	(tc, ct)	(tt, tt)

Tableau 2 : matrice des payoffs.

Rappelons que pour se trouver dans une situation similaire au dilemme du prisonnier, il est important qu'aucun des participants ne sache à priori ce que l'autre a décidé de jouer et que les inégalités suivantes soient respectées : $tc > cc > tt > ct$. Il faut également que soit respectée l'inégalité : $2 * cc > tc + ct$.

On peut déjà remarquer que pour obtenir le gain le plus important, un joueur a tout intérêt à trahir, puisque $tc > cc$ et $tt > ct$. On peut donc conclure que la coopération n'a aucun intérêt dans le dilemme simple, si on privilégie l'intérêt personnel.

Dans le « **dilemme du prisonnier simple** », la coopération est avantageuse si on privilégie l'intérêt du groupe et que les deux joueurs décident de coopérer, puisque la somme des gains des deux participants est plus élevée lorsqu'ils coopèrent tous les deux que lorsqu'un seul coopère ou que les deux décident de trahir ($cc + cc > tc + ct$), ($cc + cc > tt + tt$).

1.2.2. Le dilemme itéré

Le « **dilemme du prisonnier itéré** » consiste en une succession de « **dilemmes du prisonnier simples** » entre deux participants. On parlera alors d'un « **affrontement** » entre ces deux participants.

Comme nous l'avons vu plus haut, lors d'une « **part** » unique, le participant qui désire servir ses propres intérêts devra trahir. Ceci n'est plus valable dans le cas

du dilemme itéré, car si un participant réactif s'est fait duper lors d'une part, il y a de fortes chances qu'il se venge la part suivante et donc qu'il trahisse celui qui l'avait dupé. Si le nombre de parts est élevé, les deux participants obtiendront de très faibles scores en comparaison avec ceux qu'ils auraient obtenus s'ils avaient réussi à s'entendre et ne pas se trahir l'un l'autre.

En pratique on observe d'ailleurs souvent une coopération mutuelle entre participants à un « **dilemme itéré du prisonnier** ». Cependant, il faut que le nombre de parts pendant lesquelles s'affrontent les participants ne soit pas connu à l'avance des participants. Si ce n'était pas le cas, on se retrouverait dans la situation « **du paradoxe du pendu** » (Falletta 1985, Poundstone 1990), car les deux entités sauront qu'à la dernière « **part** » elles devront se trahir si elles cherchent à maximiser leurs gains. Si en plus ces deux entités sont **conscientes** qu'elles essaient toutes deux de maximiser leurs gains et donc trahiront toutes deux à la dernière part, elle n'ont plus d'intérêt individuel à coopérer à l'avant-dernière part, puisque cela ne servira pas à faire coopérer l'autre à la dernière part. Dès lors, l'avant dernière part devient la dernière part et ainsi de suite ; dans ces conditions, les entités rationnelles et égoïstes trahiraient donc à chaque part. Cependant, si une stratégie n'est pas certaine que la stratégie adverse la trahira la dernière part, elle n'a pas nécessairement intérêt à la trahir avant la dernière part, car la stratégie adverse pourrait se venger à la dernière part.

1.3. Différents modes de confrontations

Dans cette section, nous allons présenter les différents modes de « **confrontation** », que sont la confrontation simple, la confrontation généralisée et la confrontation écologique

1.3.1. La confrontation simple

La « **confrontation simple** » est un « **affrontement** » entre deux participants durant un certain nombre de « **coups** » ou de « **parts** », ces deux termes seront indifféremment utilisés dans la suite de l'exposé. Le résultat de cette confrontation est obtenu en additionnant les payoffs (bénéfices) obtenus lors de chacun de ces coups.

Le participant qui a obtenu le plus gros score est victorieux; cependant, en le confrontant à d'autres participants il n'est pas dit qu'il aurait toujours obtenu le plus gros score, afin de vérifier cela, il faut réaliser une confrontation généralisée.

1.3.2. La confrontation généralisée

La « **confrontation généralisée** » se fait entre plus de deux participants, des confrontations simples sont organisées entre chacun de ceux-ci deux à deux. C'est-à-dire que chaque participant affrontera tous les autres participants. Les résultats sont obtenus en additionnant les résultats de toutes les confrontations simples. En général, le participant victorieux d'une confrontation généralisée est un participant qui a pour objectif principal de faire le plus de points face aux autres, et qui n'a donc pas l'objectif de ne jamais perdre une « **confrontation simple** » puisque dans ce cas, il risque de faire de très mauvais scores (section 1.2.2. §2 p 8). Certains auteurs emploient également le terme de « **tournoi** » comme synonyme de « **confrontation généralisée** ».

1.3.3. La confrontation écologique

Dans « **la confrontation écologique** », on considère une population d'individus utilisant des « **stratégies** » diverses. Nous allons donc définir ici le terme de « **stratégie** » :

> Une stratégie adoptée par une entité peut être considérée comme **une règle ou un ensemble de règles** qui permet(tent) de déterminer (éventuellement au moyen d'un tirage au sort ou en fonction des parts jouées précédemment) ce que cette entité va jouer (trahir ou coopérer) lors d'une part.

Au début de la confrontation, on fixe un certain effectif d'individus par stratégie, celui-ci est le même pour toutes les stratégies, (rem : pour des raisons de facilité nous disons que nous fixons un effectif par stratégie alors qu'il serait plus correct de dire que des individus adoptent une stratégie donnée). Une succession de « **confrontations généralisées** » sont alors réalisées entre les individus de cette population. Les effectifs des stratégies sont recalculés entre chaque confrontation

généralisée en fonction des scores obtenus. Ainsi, les stratégies ayant les meilleurs scores verront leur effectif augmenter, tandis que celles ayant obtenu un mauvais score verront leur effectif diminuer. Il est intéressant de noter que l'évolution de l'effectif d'une stratégie (croissance ou décroissance) est dépendante de l'évolution des effectifs des autres stratégies.

1.4. Les stratégies de jeu

Dans cette section, nous allons présenter dans le cadre du « **dilemme itéré du prisonnier fini** » (section 1.2.2 p 8.) quelques stratégies types assez simples. Lorsque le dilemme est itéré, le problème ne se pose plus sous la forme "trahir ou coopérer", mais sous la forme "quelle « **stratégie** » dois-je adopter en fonction du comportement de l'entité adverse ?" .

On peut d'ores et déjà faire remarquer qu'il existe une infinité de stratégies possibles et qu'il est toujours possible de combiner plusieurs stratégies afin de former une nouvelle stratégie. Dans ce qui suit, nous évaluerons les performances de quelques stratégies en relation avec leur « **environnement stratégique** » ; c'est-à-dire les stratégies qu'elles affrontent pendant un nombre de « **parts** » défini par l'expérimentateur.

1.4.1. Présentation de quelques stratégies

Dans la suite de l'exposé, lorsque nous voudrons désigner une stratégie particulière, nous mettrons son nom en majuscule ; par exemple GENTIL est la stratégie qui coopère à chaque part. Il ne faudra pas confondre la stratégie GENTIL et une stratégie « **gentille** » qui fait partie de la classe des stratégies « **gentilles** ».

En effet, toutes les stratégies peuvent être classées en stratégies « **méchantes** » et « **gentilles** » (J. P. Delahaye & Ph. Mathieu, 1992) selon respectivement qu'il est possible ou impossible qu'elles prennent l'initiative de la trahison ; les stratégies du Tableau 3 sont classées en stratégies « **méchantes** » et « **gentilles** ».

On peut également classer les stratégies en stratégies à jeux dépendants de celui de l'adversaire comme par exemple TIT_FOR_TAT, RANCUNIER, ... ; et en stratégies à jeux indépendants de celui de l'adversaire comme : LUNATIQUE, GENTIL,... (voir la signification des stratégies Tableau 3). D'autres classements existent : stratégies probabilistes, stratégies à jeux dépendants de la réponse de l'adversaire aux jeux qu'elles ont précédemment joué,.... Ces classements peuvent aider à rapidement analyser le classement des stratégies.

On pourrait également classer les stratégies en fonction de leur complexité, mais ceci risque d'être plus compliqué, car il peut exister plusieurs niveaux de complexité, en principe une infinité. De plus, on peut adopter plusieurs définitions de cette complexité.

STRATEGIES	DESCRIPTION		
GENTIL	je coopère toujours	i	g
MECHANT	je trahis toujours	i	m
LUNATIQUE	à chaque « **part** », je choisis au hasard de coopérer ou de trahir	i	m
TIT_FOR_TAT	à la première « **part** », je coopère ensuite, je joue ce que l'autre a joué la « **part** » précédente : si il a trahi à la « **part** » N, je le trahis à la « **part** » N+1, sinon je coopère	d	g
RANCUNIER	je coopère tant que l'autre coopère, mais si il me trahit lors d'une « **part** », alors je le trahirai toujours lors des « **parts** » ultérieures	d	g
PERIODIQUE_MECHANT	je joue périodiquement : trahir, trahir, coopérer,....	i	m
PERIODIQUE_GENTIL	je joue périodiquement : coopérer, coopérer, trahir, ...	i	m
MAJORITE_MOU	je coopère tant que l'autre n'a pas trahi plus de fois qu'il n'a coopéré	d	g
MAJORITE_DUR	je coopère tant que l'autre coopère plus qu'il ne trahit, et donc à la première « **part** », je trahis	d	m
SONDEUR	sonde son adversaire en jouant (trahir, coopérer, coopérer) les trois premiers coups ; si au coup 2 et 3 l'autre a coopéré, je joue toujours MECHANT, sinon je joue TIT_FOR_TAT.	d	m
MEFIANT	je commence par trahir et ensuite je joue ce que l'autre a joué la « **part** » précédente	d	m
TIT_FOR_TAT_DUR	coopère toujours sauf si l'autre a trahi l'un des deux derniers coups et donc trahit deux fois après une trahison	d	g

Tableau 3 : caractéristiques des stratégies de l'article de J. P. Delahaye & Ph. Mathieu (1992). Dans la troisième colonne : un « i » signifie qu'il s'agit d'une stratégie à jeux indépendants de celui de son adversaire, tandis qu'un « d » signifie qu'il s'agit d'une stratégie à jeux dépendants de celui de l'adversaire. Dans la quatrième colonne : un « g » signifie que la stratégie est « **gentille** » et un « m » signifie qu'elle est « **méchante** ».

On peut remarquer qu'une stratégie qui coopère plus qu'elle ne trahit peut être classée dans les stratégies « **méchantes** », ainsi une stratégie qui trahirait la première fois et qui coopérerait toujours ensuite serait considérée comme « **méchante** » !

1.4.2. Les stratégies et leur environnement stratégique

La stratégie MECHANT ne perd contre aucune stratégie, elle bat son adversaire ou fait un score identique à ce dernier, mais lorsque cette stratégie est confrontée à elle-même ou à des « **stratégies réactives** » comme TIT_FOR_TAT ou RANCUNIER elle réalise des scores médiocres. Par « **stratégies réactives** », nous sous-entendons des stratégies qui réagissent par un certain nombre de trahisons en réponse à une trahison lors d'une part précédente. Dans le pire des cas, ces dernières obtiennent des scores à peine inférieurs aux stratégies qu'elles affrontent, mais en général, lors d'une « **confrontation généralisée** », elles obtiennent de beaux scores. C'est ce qui explique que souvent la stratégie MECHANT se retrouve classée derrière les stratégies RANCUNIER et TIT_FOR_TAT lorsque l'on organise un « **tournoi** » avec une proportion suffisamment élevée de stratégies réactives.

On peut déjà faire remarquer qu'il n'existe pas de stratégie qui puisse se classer première ou première ex aequo dans n'importe quel environnement stratégique ; en effet, si elle existait, elle devrait trahir en premier pour être certaine de faire un score au moins égal à celui de MECHANT, et si elle trahit en premier, elle ne fera qu'un faible score face à une RANCUNIER et pourrait se retrouver classée derrière une GENTIL si la proportion de RANCUNIER est suffisamment élevée dans l'environnement stratégique qu'elle affronte.

Dans ce qui suit, nous discutons de quelques résultats obtenus par différents auteurs en réalisant des tournois entre les stratégies, c'est-à-dire en confrontant deux à deux chacune des stratégies d'un ensemble donné de stratégies ; ces tournois portent le nom de confrontations généralisées (voir section 1.3.2. p 10).

1.4.2.1. Confrontations généralisées de stratégies

Dans une première expérience qui mettait en jeu 63 stratégies très variées, R. Axelrod (1984) a pu mettre en évidence qu'en général c'était la stratégie TIT_FOR_TAT qui faisait les plus beaux scores. De plus, les stratégies « **méchantes** » (pouvant prendre l'initiative de la trahison) étaient presque toujours mal classées, alors que les « **gentilles** » (qui ne prennent jamais l'initiative de la trahison) étaient presque toutes bien classées. TIT_FOR_TAT contraint son adversaire à coopérer si il veut obtenir des points, c'est probablement cette réactivité qui est la cause principale du succès de TIT_FOR_TAT. R. Axelrod (1984) a retiré quelques enseignements de ses expériences :

- il vaut mieux être « **gentil** » que « **méchant** »
- il est nécessaire d'être réactif et donc de punir l'autre si il nous a trahi
- il faut se réconcilier rapidement
- il ne sert à rien de ruser car la clarté du comportement est ce qui est le plus susceptible d'instaurer une coopération mutuelle prolongée et donc profitable

Ultérieurement J. P. Delahaye & Ph. Mathieu (1992) ont réalisé des expériences dont les résultats semblent confirmer ceux obtenus par R. Axelrod (1984) en ce qui concerne la supériorité de TIT_FOR_TAT. Lors d'une confrontation généralisée des 12 stratégies du Tableau 3 (p 13) durant 1000 « **parts** » avec les coefficients suivants : ct=0 (je coopère l'autre me trahit), tt=1, cc=3, tc=5 (je trahis l'autre coopère) (Tableau 2, p 8). Même en changeant la valeur des coefficients et le nombre de « **parts** », pour autant que les inégalités soient respectées et que le nombre de « **parts** » soit assez élevé, TIT_FOR_TAT est souvent la mieux classée (Tableau 4, p 16) et les stratégies de tête sont le plus souvent des stratégies ne prenant pas l'initiative de la trahison donc des « **gentilles** ». J. P. Delahaye & Ph. Mathieu (1992) remarquent également qu'en-dessous de 4 « **parts** » les stratégies « **méchantes** » l'emportent dans une confrontation généralisée des 12 stratégies du Tableau 3 (p 13), et qu'au-dessus de 4 ce sont les « **gentilles** ».

STRATEGIES		SCORES
TIT_FOR_TAT	g	30890
MAJORITE_MOU	g	30527
RANCUNIER	g	28045
SONDEUR	m	27507
PERIODIQUE_GENTIL	m	27320
TIT_FOR_TAT_DUR	g	27309
GENTIL	g	25506
LUNATIQUE	m	24336
MEFIANT	m	22925
MAJORITE_DUR	m	22066
MECHANT	m	22022
PERIODIQUE_MECHANT	m	21210

Tableau 4: scores obtenus par les stratégies lors d'une confrontation généralisée de ces stratégies dans « **un dilemme du prisonnier itéré fini** » avec les coefficients : ct=0 (je coopère l'autre me trahit), tt=1, cc=3, tc=5 (je trahis l'autre coopère) ; chaque stratégie affronte pendant 1000 « **parts** » les 11 autres stratégies, « g » veut dire qu'il s'agit d'une stratégie « **gentille** » et « m » d'une « **méchante** ».

1.4.2.2. Concours de stratégies imaginées chacune par une personne différente

Afin de voir si il existait d'autres stratégies faisant des meilleurs scores que TIT_FOR_TAT dans un environnement stratégique imaginé par plusieurs personnes, J. P. Delahaye & Ph. Mathieu (1993) ont lancé un concours où les participants ont proposé chacun une seule stratégie et ceci afin d'éviter les tricheries ; il est en effet aisé de faire gagner une stratégie "maître" si cette dernière joue contre un nombre suffisant de stratégies "esclaves" destinées uniquement à faire augmenter le score de la stratégie "maître".

Dans ce concours, les stratégies s'affrontent toutes pendant 1000 « **parts** », les coefficients précédents ont été repris (ct=0, tt=1, cc=3, tc=5) (section 1.4.2.1, p 15), et il est également permis de renoncer à jouer contre un adversaire, mais dans ce cas, celui qui renonce devra le faire jusqu'à la fin ainsi que l'adversaire. Le renoncement (rr=2) rapporte donc plus qu'une trahison mutuelle mais moins qu'une coopération mutuelle ; ce coefficient n'encourage pas à jouer le renoncement suite à la première trahison puisqu'une suite de tc, ct, tc, ct, tc, ct, ... rapporte en moyenne 2,5 points alors que le renoncement à jouer ne rapporte que deux points en moyenne par « **part** ». J. P. Delahaye & Ph. Mathieu (1993) expliquent qu'ils ont imposé le

renoncement définitif afin d'éviter d'avoir à chaque étape la possibilité de choisir entre trois options, car jusqu'à présent rien de très clair n'a été obtenu sur de tels jeux même si ils représentent bien (mieux) certaines situations réelles.

1.4.2.3. Test de la robustesse des stratégies

Dans leur expérience de 1993 où 95 stratégies ont participé à une confrontation généralisée, J. P. Delahaye & Ph. Mathieu ont identifié une stratégie qu'ils ont qualifiée de « **robuste** ». Celle-ci était la mieux classée des 95, pour cette raison, ils l'ont appelée LA_MEILLEURE.

J. P. Delahaye & Ph. Mathieu (1993) ont testé cette robustesse en réalisant une « **simulation écologique** » ; ils ont composé une première génération en attribuant à chaque stratégie un effectif de 100 entités, le nouvel effectif de chaque stratégie à la génération suivante est déterminé en fonction des scores qu'elles obtiennent après une confrontation généralisée. Dans ces conditions les stratégies « **méchantes** » qui obtiennent un faible score sont rapidement éliminées, car les stratégies du tournoi sont des stratégies éditées par des humains, et que la plupart d'entre elles réagissent aux trahisons dont elles font l'objet soit en trahissant, soit en renonçant à jouer.

De même, les stratégies qui avaient un avantage sur les autres parce qu'elles perdaient moins de points face à ces « **méchantes** » disparaissent ou obtiennent de très mauvais scores, puisqu'elles ne possèdent plus cet avantage sur d'autres stratégies plus gentilles. Ils ont ainsi observé que la deuxième stratégie du classement initial se retrouve éliminée après quelques générations.

1.4.2.4. Commentaires sur le terme de robustesse

Nous ne pensons pas que cette **simulation écologique** constitue une preuve dans l'absolu de la « **robustesse** » de la stratégie LA_MEILLEURE puisque cette simulation semble conduire à la création d'un environnement gentil, or il pourrait également exister des environnements essentiellement méchants dans lesquels cette stratégie ne soit plus la meilleure et où elle obtiendrait même de mauvais scores, il faudrait donc la tester dans ces environnements. En réalisant une simulation écologique avec de tels environnements, il se pourrait également que la stratégie LA_MEILLEURE disparaisse.

selon nous :

une stratégie est « **robuste** » si lors d'une confrontation généralisée ou écologique, elle est bien classée quel que soit l'environnement stratégique affronté.

Nous pensons qu'il serait possible d'évaluer la « **robustesse** » d'une stratégie en la confrontant à un ou plusieurs environnements stratégiques. Si ces environnements sont bien choisis, on peut penser qu'il y a plus de chances que cette stratégie obtienne de bons scores ou gagne une confrontation écologique dans un autre environnement. Cependant, on ne pourra jamais certifier qu'une stratégie est « **robuste** » dans l'absolu.

La confrontation généralisée de J. P. Delahaye & Ph. Mathieu (1993) impliquant 95 stratégies chacune éditée par un auteur différent, on peut considérer que ces stratégies constituent un environnement stratégique plausible dans le monde réel des humains. Mais, idéalement, pour tester la « **robustesse**» de la stratégie LA_MEILLEURE dans un environnement constitué de stratégies adoptées par des humains, il faudrait réaliser plusieurs concours similaires à celui de J. P. Delahaye & Ph. Mathieu (1993) dans des pays et des milieux sociaux différents. Même dans ce cas, la robustesse est limitée à des types d'environnement stratégiques humains.

La « **robustesse** » d'une stratégie, telle que sous-entendue par J. P. Delahaye & Ph. Mathieu (1993) (section 1.4.2.3, p 17), est contradictoire avec ce que l'on observe

dans le monde vivant ; en effet, on observe qu'en général une espèce très bien adaptée à son environnement est souvent condamnée à disparaître si des changements même minimes apparaissent dans son environnement (exemple : beaucoup d'espèces animales australiennes ont disparu suite à l'introduction de petits carnivores européens), un peu comme la stratégie classée la deuxième du concours qui disparaissait dans une simulation écologique. Si l'on considère (peut-être à tort) que l'environnement stratégique constitué par l'ensemble des stratégies humaines est peu changeant, la « **robustesse** » d'une stratégie restreinte à un tel environnement ne serait plus en contradiction avec ce qui est observé dans le monde vivant.

1.4.2.5. Principales caractéristiques de la stratégie LA_MEILLEURE

- Elle coopère à la première « **part** »
- Elle ne prend jamais l'initiative de la trahison : c'est une « **gentille** »
- Elle renonce si après 20 parts le score obtenu est inférieur à 1,5
- Elle entre dans une période de punition lorsqu'elle est trahie
- Elle est de plus en plus sévère
- Elle tente de calmer son adversaire après une période de punition en coopérant deux fois de suite
- Elle est compréhensive car elle ne tient pas compte des trahisons de l'adversaire pendant qu'elle est en train de le punir

Il semble que la dernière caractéristique soit un défaut ; en effet, J. P. Delahaye & Ph. Mathieu (1993) ont réalisé une variante de cette stratégie qui prend également en compte les trahisons de l'adversaire pendant les phases de punition et cette stratégie obtenait un meilleur score que LA_MEILLEURE.

J. P. Delahaye & Ph. Mathieu (1993) pensent que certaines stratégies sont bonnes et d'autres mauvaises ; ils pensent également que la combinaison de certaines stratégies peut mener à de meilleures stratégies mais plus complexes, ainsi les stratégies utilisant l'idée du SEUIL et du TIT_FOR_TAT (donnant-donnant) sont assez bien classées entre la 7ème et 47ème place. Ceci contredit ce que disait R. Axelrod (1984) à propos de l'avantage qu'a une stratégie à avoir un comportement clair et donc simple.

J. P. Delahaye & Ph. Mathieu (1993) insistent également sur l'intérêt du renoncement puisque dans les 40 premières stratégies seules 3 ne l'utilisent pas, les 26 autres n'utilisant pas le renoncement sont classées après la 40ème position. Seules deux stratégies parmi les 40 premières prennent l'initiative de trahir. Lorsque l'on ajoute à ces 95 stratégies les 12 de l'article précédent (J. P. Delahaye & Ph. Mathieu, 1992), il y a peu de changement et la meilleure des 12 est RANCUNIER. J. P. Delahaye & Ph. Mathieu (1992) pensent également qu'il n'y a pas de limite au perfectionnement de la stratégie TIT_FOR_TAT ; ils ont d'ailleurs réussi à créer une stratégie meilleure que LA_MEILLEURE. Ceci ne constitue cependant pas une preuve du perfectionnement potentiellement infini de TIT_FOR_TAT ; de plus, les auteurs parlent toujours du perfectionnement de TIT_FOR_TAT ; pourquoi pas une autre stratégie ? On dirait qu'ils croient en TIT_FOR_TAT.

1.4.2.6. Une stratégie « simple » meilleure que TIT_FOR_TAT

En modifiant petit à petit les paramètres des stratégies des différents concours qu'ils ont organisés, J. P. Delahaye & Ph. Mathieu (1996) ont réussi à trouver une stratégie obtenant de meilleurs scores que TIT_FOR_TAT dans bien des environnements stratégiques. Ils ont nommé cette stratégie GRADUELLE. A chaque fois que GRADUELLE est trahie par son adversaire, elle regarde dans le passé combien de fois cet adversaire l'a trahie, si son adversaire l'a trahie N fois, elle le trahit N fois de suite et coopère ensuite deux fois de suite. C'est une stratégie fort semblable à la stratégie LA_MEILLEURE qui tient compte de ses qualités principales : sa période de punition et sa réactivité; cependant, elle n'utilise pas le renoncement et tient compte **de toutes les trahisons** de l'adversaire même si il a trahi pendant la période de punition.

J. P. Delahaye & Ph. Mathieu, (1996) pensent qu'une stratégie ne doit pas spécialement avoir un comportement clair comme l'a signalé R. Axelrod (1984), car l'environnement stratégique n'est pas toujours constitué de stratégies capables de comprendre le comportement stratégique d'une autre, de plus certaines stratégies ne sont pas rationnelles et ne cherchent pas spécialement à obtenir le plus de points possibles : LUNATIQUE, MECHANT, ... On peut également penser au vu de la

supériorité de GRADUELLE sur TIT_FOR_TAT qu'il est important de ne pas pardonner rapidement à son adversaire.

J. P. Delahaye & Ph. Mathieu (1993) suggèrent l'utilisation des algorithmes génétiques pour tenter d'obtenir des stratégies « **robustes** » probablement qu'ils veulent dire « **robuste** » dans l'environnement stratégique humain. C'est cette méthode que nous avons utilisée dans la partie expérimentale de ce mémoire, avant d'aborder cette dernière, nous allons présenter la technique des algorithmes génétiques imaginée par Holland (1975) et adaptée au cas du « **dilemme itéré du prisonnier fini** » par R. Axelrod (1987).

Chapitre II : LES ALGORITHMES GENETIQUES

Nous allons tout d'abord rappeler quelques notions de Génétique et de la Théorie de la Sélection Naturelle qui ont permis de générer la technique des algorithmes génétiques.

2.1. Notions de Génétique et de la Théorie de la Sélection Naturelle

Chez les vivants, un chromosome détermine en grande partie les caractéristiques physiques et comportementales (particulièrement instinctives) de l'organisme qui le possède et est transmis à la descendance de cet organisme.

Les chromosomes contiennent les gènes de l'individu ; en simplifiant, on peut également dire que les gènes représentent le code d'une caractéristique de l'individu. L'ensemble des chromosomes d'un individu constitue son matériel génétique et ce matériel génétique diffère très peu entre individus d'une même espèce ou d'espèces proches. Chez les individus à reproduction **sexuée**, on a généralement plusieurs paires de chromosomes, un chromosome d'une paire provient de la mère et l'autre du père ; ces chromosomes appariés sont appelés homologues car ils contiennent des gènes qui sont le code d'une même caractéristique, ces gènes diffèrent toutefois légèrement l'un de l'autre et généralement un seul des deux gènes codant pour une même caractéristique s'exprime ; c'est l'allèle dominant, tandis que l'autre est appelé allèle récessif.

Selon la Théorie de la Sélection Naturelle, certains êtres vivants sont mieux adaptés que d'autres à l'environnement dans lequel ils évoluent, et auront plus de chance de se reproduire que d'autres ; leurs gènes peuvent être transmis à leur descendance de manière **sexuée** ; dans ce cas, dans la cellule à l'origine des gamètes ont lieu des crossing-over qui sont des échanges de gènes entre les deux chromosomes homologues, suite à cela cette cellule se divise en deux cellules ayant chacune un des deux chromosomes comportant à la fois des gènes du père et de la mère et appelées gamètes.

2.2. Introduction aux algorithmes génétiques

Les algorithmes génétiques représentent une famille assez riche et très intéressante d'algorithmes d'optimisation stochastique ; ils sont fondés sur les mécanismes de la Sélection Naturelle et de la Génétique.

Les champs d'application sont fort diversifiés ; on retrouve les algorithmes génétiques aussi bien en théorie des graphes qu'en compression d'images numérisées ou encore en programmation automatique (Goldberg, 1983), (Suh, Gucht, 1987). Leur principe est d'opérer une recherche stochastique sur un important espace à travers un ensemble de pseudo-solutions.

Les termes utilisés dans le domaine des algorithmes génétiques n'ont qu'une faible analogie avec ceux employés en biologie. De plus, ils peuvent désigner des mécanismes différents selon les auteurs qui les emploient. Pour cette raison, nous expliquerons ce que des termes comme crossing-over et croisement désignent dans notre programme (section 3.2, p 31).

Le principe des algorithmes génétiques est de faire évoluer une population dans un environnement donné, de telle manière qu'apparaissent dans cette population des individus bien adaptés à leur environnement. Ces derniers étant bien adaptés auront plus de chance de se reproduire, d'élever leur progéniture avec succès, ..., et donc de devenir majoritaires dans cette population. Ce principe étant tiré des lois de la Sélection Naturelle formulées par Darwin.

On parle de « **fitness** » pour la fonction d'évaluation (fonction positive) de l'adaptation d'un individu à son environnement. Ainsi, les individus ayant un « **fitness** » important ont plus de chance d'être sélectionnés pour produire la génération suivante. Dans notre programme, cette fonction est tout simplement le score obtenu par un individu après une confrontation généralisée, dans son environnement ; on réalise alors une sélection des individus qui se reproduiront proportionnellement aux scores qu'ils ont obtenus.

Il ressort clairement de divers travaux que le choix des valeurs numériques pour les différents paramètres (crossing-overs, nombre d'individus sélectionnés pour la reproduction, ...) reste une tâche assez complexe (I.C.Lerman, R.F.Ngouenet).

2.3. Adaptation de l'algorithme génétique au « dilemme du prisonnier »

R. Axelrod (1987) utilisa la technique des algorithmes génétiques pour étudier le « **dilemme itéré du prisonnier fini** ». Chaque stratégie est représentée par un « **chromosome** » qui est une suite de « **parts** » (gènes) lors desquelles la stratégie joue : soit coopérer, soit trahir. Comme précédemment, les performances d'une stratégie sont jugées dans un environnement stratégique donné.

Le programme de simulation de R. Axelrod (1987) travaille en cinq étapes :

1. Une population initiale est choisie dans laquelle chaque individu adopte une stratégie représentée par un chromosome qui est une chaîne de C (coopérer) et de T (trahir) répartis au hasard sur cette chaîne.

2. Chaque individu effectue une confrontation généralisée contre 8 « **stratégies environnementales** » fixées, et leur score est la moyenne de tous les scores obtenus

3. Les individus très performants dont le score est un écart type au-dessus de la moyenne peuvent se reproduire « sexuellement » deux fois ; les individus moyens peuvent se reproduire une fois, tandis que les individus dont le score est inférieur à la moyenne moins un écart type, ne peuvent se reproduire.

4. Les individus se reproduisent alors aléatoirement entre eux pour produire deux descendants dont les chromosomes respectifs résultent d'un crossing-over entre ceux des deux parents. Par exemple si un parent a un chromosome constitué de 10 gènes C et que l'autre en a un constitué de 10 gènes T alors si un crossing-over survient après le troisième gène, un des descendant recevra 3 C et 7 T et l'autre 3 T et 7 C. Il peut y avoir plus d'un

crossing-over par reproduction. Les mutations ont lieu en changeant au hasard une très petite proportion de C en T ou inversement.

5. Ceci donne une nouvelle génération d'individus dont les stratégies respectives ressemblent plus aux stratégies de la génération précédente ayant obtenu de bons scores face à l'environnement stratégique puisqu'elles se sont mieux reproduites que les stratégies ayant obtenu un mauvais score.

Suite à 40 simulations effectuées sur une population de vingt individus (chacun ayant adopté une stratégie particulière) qui affrontent 8 stratégies environnementales pendant 151 « **parts** » (ce qui fait 24 000 « **parts** » par génération) et après 50 générations, la plupart des nouvelles stratégies ressemblent à TIT_FOR_TAT, avec l'acceptation de se faire duper (ct), si la « **part** » avant, on avait trahi.

R. Axelrod pense que dans un environnement tout différent, les nouvelles stratégies meilleures que TIT_FOR_TAT ne seraient probablement pas apparues, et qu'elles auraient même obtenu des mauvais scores. Ceci est en accord avec l'observation faite par J. P. Delahaye & Ph. Mathieu (1993) à propos de la deuxième stratégie de leur confrontation généralisée qui disparaît rapidement lors d'une simulation écologique car les méchantes dont elle profite, disparaissent (section 1.4.2.3, p 17). Cependant, J. P. Delahaye & Ph. Mathieu (1993) ont fait remarquer que la stratégie LA_MEILLEURE (section 1.4.2.5, p 19) semblait « **robuste** », il serait donc intéressant de tester si les algorithmes génétiques nous permettraient de découvrir des stratégies « **robustes** ». L'idéal, serait donc d'essayer tout d'abord de sélectionner des bonnes stratégies apparues dans un environnement donné et ensuite de tester leur résistance à la variation de leur environnement.

J. P. Delahaye & Ph. Mathieu (1996) ont également essayé d'améliorer la stratégie GRADUELLE (section 1.4.2.6, p 20) en employant les algorithmes génétiques et en utilisant comme gènes des paramètres comme par exemple :

- Aléa : un nombre aléatoire pour déterminer à quelle part, on trahit
- Vision : la longueur du passé qu'une stratégie peut connaître
- Evolution de la punition : si la punition évolue de manière exponentielle ou logarithmique
- ...

Ils ont réussi à trouver une meilleure stratégie que GRADUELLE, ce qui va dans le sens que l'amélioration que l'on peut apporter aux stratégies est potentiellement infinie, du moins en ce qui concerne sa « **robustesse** ».

2.4. Voies de recherche suivies

Nous avons estimé que la technique des algorithmes génétiques était intéressante pour la propriété qu'elle a de trouver rapidement une stratégie adaptée à son environnement stratégique (R. Axelrod, 1987). Cependant, nous avons voulu modifier la méthode employée par R. Axelrod (section 2.2, p 23) qui, rappelons le, consistait à considérer chaque gène d'une « **stratégie** » ou d'un « **chromosome** » comme un C ou un T ; c'est-à-dire le jeu joué lors d'une « **part** » par un individu ayant adopté cette « **stratégie** ». Dans ces chromosomes, il n'y a donc que deux gènes différents possibles et l'emplacement du gène dans le chromosome correspond au numéro de « **part** » joué.

2.4.1. Définition des gènes utilisés dans notre expérience

Dans notre expérience, nous considérons un « **gène** » plutôt comme un « **comportement** » qui est adopté lors d'une « **part** » (TIT_FOR_TAT, MECHANT, GENTIL, ...). Le nombre de ces gènes est égal au nombre de « **parts** » pendant lesquelles on désire faire s'affronter deux entités (_déjà été discuté à la section 2.4.2, p 27.

Figure 1, p 30). La stratégie adoptée par une entité pendant l'affrontement peut donc être considérée comme une suite de « **comportements** » adoptés lors de l'affrontement ; si ces « **comportements** » sont différents l'un de l'autre, nous disons que la « **stratégie** » est « **hybride** », sinon nous disons qu'elle est « **simple** ».

Les gènes d'une « **stratégie simple** » adoptent tous le même « **comportement** », nous nous permettrons d'appeler une « **stratégie simple** » par le nom de ce comportement.

Au début de l'expérience, l'utilisateur a la possibilité de définir un « **environnement stratégique** » composé de « **stratégie(s) simple(s)** » ; ces stratégies seront appelées « **stratégies environnementales** » (déjà été discuté à la section 2.4.2, p 27.

Figure 1, p 30). L'environnement stratégique est ainsi fixé pour toute la durée de l'expérience.

L'utilisateur détermine d'autres « **stratégies simples** » que nous avons appelées : « **stratégies à tester de départ** ». Ces stratégies ainsi que les descendantes de ces stratégies auront la possibilité de s'échanger entre elles leurs gènes par « **crossing-over** », ces deux types de stratégies sont également appelées « **stratégies à tester** ». Suite à l'affrontement d'un environnement stratégique donné, et en fonction des points qu'elles obtiendront face à cet environnement, certaines stratégies vont se reproduire entre elles de manière « **sexuée** » pour donner parfois naissance à des « **stratégies hybrides** ».

2.4.2. Intérêt d'avoir considéré les gènes comme des comportements

Les jeux de nos « **stratégies** » sont non fixés, puisqu'à chaque part la stratégie joue ce que dicte le « **comportement** » du gène correspondant, et pas d'office un C ou un T comme les jeux des stratégies de R. Axelrod. De cette façon, il est possible de jouer de façon réactive au jeu joué par l'adversaire.

Suite à l'évolution dans un environnement stratégique donné, on peut donc obtenir des « **stratégies hybrides** » qui obtiennent de meilleurs scores face à cet environnement que les stratégies d'affrontement initialement fixées par l'expérimentateur.

Dans les expériences de R. Axelrod, il est quasi impossible d'obtenir des stratégies « **robustes** » même si on limite cette robustesse à l'environnement des stratégies humaines, puisque les meilleures stratégies apparues grâce à la méthode des algorithmes génétiques ont un jeu fixé pour chaque part avant même d'affronter une stratégie quelconque. Une possibilité serait d'analyser les jeux joués par ces stratégies en fonction du jeu joué par les « **stratégies environnementales** » (section 2.3, (2ème

étape, p 24)) grâce auxquelles elles sont apparues et d'essayer de découvrir ce qui fait le succès de ces stratégies pour essayer de créer une nouvelle stratégie « **robuste** ».

C'est principalement pour cette raison que nous avons préféré considérer les gènes comme des stratégies plutôt que comme des C ou des T. Nous espérions de cette manière trouver une stratégie bien adaptée à plusieurs environnements.

Nous avions également envie de permettre l'examen plus facile de la séquence de gènes d'une « **stratégie hybride** » et voir si des séquences de gènes ne se retrouvaient pas quasi systématiquement dans ces stratégies.

Il serait probablement plus intéressant d'évaluer la « **robustesse** » des stratégies en changeant l'environnement auquel elles sont confrontées, puisqu'il n'existe probablement pas de stratégies « **robustes** » dans l'absolu.

Nous avons donc conçu des environnements stratégiques types composés uniquement :

- De « **stratégies simples** » ne prenant jamais l'initiative de la trahison
- De « **stratégies simples** » pouvant prendre l'initiative de la trahison
- De « **stratégies simples** » à jeux dépendants de celui de l'adversaire
- De « **stratégies simples** » à jeux ne dépendants pas de celui de l'adversaire
- De « **stratégies simples** » à jeux dépendants de la réponse de l'adversaire à leurs jeux
- De Toutes les « **stratégies simples** » de notre programme

De plus, l'expérimentateur aura la possibilité de définir un nouvel environnement, en sélectionnant lui-même des « **stratégies simples** » parmis celles proposées.

2.4.3. Justification de la non-utilisation des mutations

Nous n'avons pas jugé nécessaire d'utiliser dans notre programme les « **mutations** » comme souvent le font les auteurs employant la technique des algorithmes génétiques. Lors d'une mutation, un gène est remplacé par un autre, ce qui permet de faire apparaître des gènes au sein des chromosomes desquels ils sont

absents. Cette propriété est intéressante, si suite à la sélection, certains gènes disparaissent et ne sont plus représentés dans aucun chromosome. Cependant, nous avons pris soin de ne jamais éliminer les « **stratégies à tester de départ** » de telle sorte que l'on conserve tous les gènes de départ. Cependant, si le nombre de stratégie(s), ou l'effectif d'individu(s) ayant adopté une stratégie augmente fortement, il se peut que la probabilité qu'ont les « **stratégies à tester de départ** » de se croiser avec d'autres, devienne très faible, et que donc leurs gènes ne se retrouvent que très peu représentés dans la population.

CHAPITRE III : DESCRIPTION DU LOGICIEL

3.1. Représentation génétique des stratégies

Une représentation schématique des chromosomes ou « **stratégies** » utilisées dans nos algorithmes génétiques est donnée à la Figure 1; l'intérêt d'une telle représentation a déjà été discuté à la section 2.4.2, p 27.

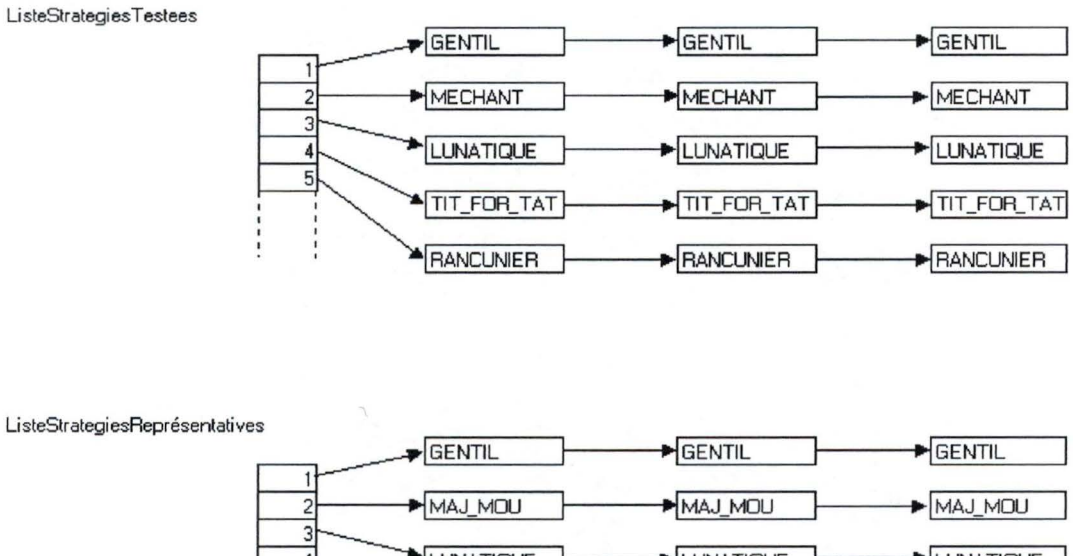

Figure 1 : Représentation des « **stratégies** » dans notre programme ; dans cet exemple, l'expérimentateur a décidé d'organiser des confrontations de 3 « **parts** » il y a donc 3 gènes par « **stratégie** », ListeStrategiesTestees et ListeStrategiesRepresentatives sont deux tableaux de pointeurs faisant respectivement les liens vers des « **stratégies à tester** » et des « **stratégies environnementales** » (les chiffres dans les cases sont les indices de celles-ci). On peut remarquer que l'utilisateur a introduit dans l'environnement deux stratégies RANCUNIER.

3.2. Présentation générale du programme

Dans cette section, nous nous contenterons de présenter le programme sans aborder l'interface, qui sera abordée à la section suivante. De plus, nous ne présenterons de façon détaillée que les modules les plus intéressants.

Le programme réalisé comporte deux grandes phases que sont: la génération de nouvelles stratégies grâce aux algorithmes génétiques, et la confrontation écologique entre stratégies. Les différents modules utilisés pour ces deux tâches peuvent être classés en modules de coordination, modules de données, et modules algorithmiques.

3.2.1. Description de la phase de génération de nouvelles stratégies

Cette phase du programme a pour but de former des nouvelles « **stratégies hybrides** ». Ces stratégies seront issues de croisements réalisés entre une série de « **stratégies à tester** ».

Pour ceci, l'utilisateur détermine

1. Les stratégies :
    - les stratégies utilisées, « **stratégies à tester de départ** » et « **stratégies environnementales** »,
    - leur nombre respectif,
2. Les paramètres de l'expérience :
    - le nombre de « **coups** », qui correspond en fait au nombre de gènes de ces stratégies,
    - le nombre de croisements, c'est-à-dire le nombre de générations successives à former,
    - le nombre de reproduction sexuées, c'est-à-dire le nombre de paires de descendants formées à chaque génération (croisement), sachant qu'une paire de descendants est formée par crossing-over de deux parents,
    - le nombre de crossing-over lors d'une reproduction,

- et le nombre de tests effectués, ou le nombre de fois que l'algorithme génétique sera lancé, tout en laissant les autres paramètres constants. De par le principe des algorithmes génétiques (section), les résultats de deux tests lancés avec les mêmes paramètres donneront très rarement les mêmes résultats.

Lors de chaque génération, les « **stratégies à tester** », sont confrontées individuellement aux « **stratégies environnementales** ». Les reproductions sont ensuite réalisées principalement entre les stratégies qui ont obtenu les meilleurs scores. Les nouvelles stratégies formées sont alors ajoutées aux « **stratégies à tester** », et on effectue une nouvelle génération (croisement).

A la fin du test, la stratégie qui a obtenu le meilleur score lors de la confrontation faisant suite au dernier croisement, est désignée : « **la meilleure stratégie du test** ».

Après avoir réalisé tous les tests, la stratégie qui a obtenu le plus gros score parmi les « **meilleures stratégies des tests** » est désignée comme « **la meilleure stratégie de l'expérience** ». De même, une nouvelle stratégie est créée en reprenant pour chaque position, le gène le plus représenté chez les « **meilleures stratégies des tests** » à cette position. Cette stratégie sera appelée « **stratégie moyenne** ».

3.2.2. Description de la phase de confrontation écologique

Cette phase du programme est une confrontation écologique tel qu'expliqué en section 3.2.3.2, p 35. Les stratégies utilisées pour cette confrontation, sont les « **stratégies environnementales** » désignées par l'utilisateur, ainsi que « **la meilleure stratégie** », ou la « **stratégie moyenne** » obtenues lors de la phase de génération de nouvelles stratégies.

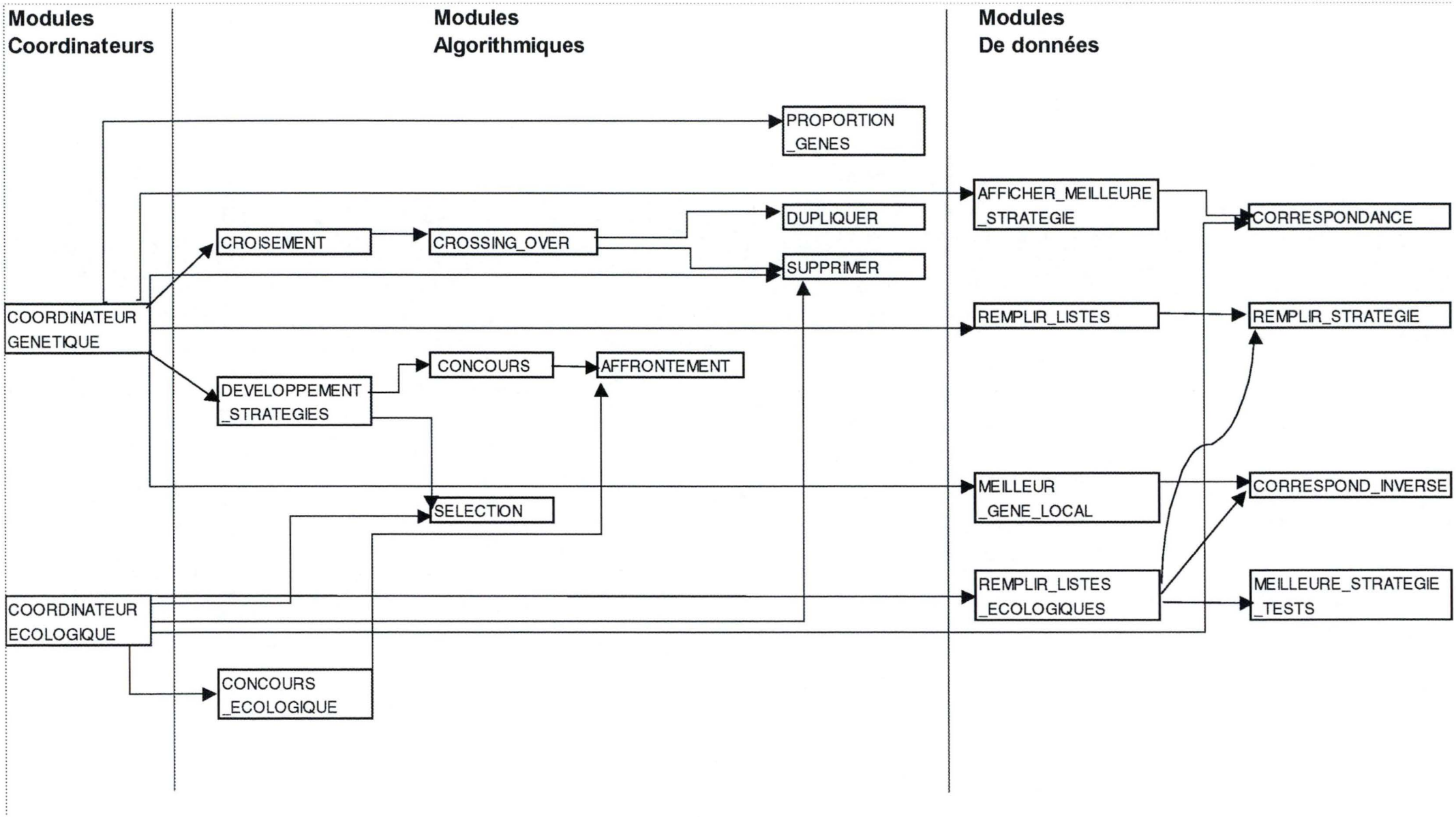

Figure 2 : Modules du programme

3.2.3. Modules de coordination

Nous retrouvons deux modules de coordination, qui correspondent aux deux phases distinctes du programme. Nous appellerons donc ces modules : module COORDINATEUR GENETIQUE, et module COORDINATEUR ECOLOGIQUE.

3.2.3.1. Module COORDINATEUR GENETIQUE

Ce module a pour but de réaliser la phase de génération de nouvelles stratégies, c'est-à-dire lancer pour chaque test le nombre de croisements désirés par l'utilisateur.

Modules de données utilisés :

- module REMPLIR_LISTES, qui se charge de remplir les listes de « **stratégies environnementales** » en début d'expérience, ainsi que les « **stratégies à tester** » avant chaque test. La structure de ces listes est représentée à la Figure 2 (p 33). Ce module se charge également de mettre ou remettre les effectifs des stratégies à la même valeur en début de test.
- module SUPPRIMER, afin de vider les listes de stratégies à tester et représentatives.
- module AFFICHER_MEILLEURE_STRATEGIE, qui permet l'affichage des « **des meilleures stratégies des tests** ».
- module PROPORTION_GENES, qui permet l'affichage des proportions respectives de chaque « **comportement** », dans les « **meilleures stratégies des tests** ».
- module MEILLEUR_GENE_LOCAL, qui permet d'afficher la « **stratégie moyenne** » (section 3.2.1, p 31).

Modules algorithmiques activés afin de réaliser les croisements :

- module DEVELOPPEMENT_STRATEGIES, qui permet de calculer le score de chaque « **stratégie à tester** » avant de déterminer celles qui seront sélectionnées pour se reproduire lors de chaque génération, ainsi qu'après le dernier CROISEMENT afin de déterminer « **la meilleure stratégie du test** ».
- module CROISEMENT, qui permet de former des nouvelles stratégies par reproduction(s) de paire(s) de stratégies.

3.2.3.2. Module coordinateur écologique

Ce module a pour but de réaliser la phase de « **confrontation écologique** ». Cette confrontation se fera sur une durée de 500 périodes, avec modification des effectifs après chacune de celles-ci.

Modules de données utilisés :

- module REMPLIR_LISTES_ECOLOGIQUES, qui se charge de remplir la liste de stratégies utilisées avant la confrontation. Sont reprises dans cette liste : les « **stratégies environnementales** » et soit la « **meilleure stratégie de l'expérience** » soit la « **stratégie moyenne** »
- module CORRESPONDANCE, qui fait la correspondance entre les « **stratégies simples** » choisies par l'utilisateur et la représentation de leur unique « **comportement** » dans le programme.
- module SUPPRIMER, afin de vider la liste des stratégies après l'affrontement.

Modules algorithmiques activés :

- CONCOURS_ECOLOGIQUE, qui permet d'effectuer les confrontations lors de chaque période.
- SELECTION, qui permet de calculer les nouveaux effectifs de chaque stratégie à chaque période.

3.2.4. Modules algorithmiques

3.2.4.1. Module DEVELOPPEMENT_STRATEGIES

Ce module se contente d'activer les modules algorithmiques CONCOURS et SELECTION.

Ce module était prévu afin de permettre de développer autant de générations que l'on voulait avant d'effectuer un CROISEMENT pour permettre de faire varier les chances qu'avaient les « **stratégies à tester** » de se croiser en fonction de leur adaptation à l'environnement stratégique. De cette façon, en augmentant le nombre de générations, les stratégies bien adaptées à leur environnement stratégique obtiennent des scores de plus en plus importants, alors que les stratégies mal adaptées voient leur score décroître, or dans notre programme, plus une stratégie a des beaux scores et plus elle a de chance d'être choisie pour être croisée avec une autre stratégie.

Pour éviter de compliquer notre interface, et pour dans un premier temps limiter les tests nous avons décidé de développer automatiquement une seule génération avant chaque CROISEMENT. Cependant, il serait peut-être intéressant de faire varier ce nombre de génération(s) afin de trouver plus rapidement des stratégies bien adaptées à leur environnement stratégique, avec l'augmentation de leur effectif avant chaque CROISEMENT, il est fort probable que les « **stratégies** » résultant des CROISEMENT soient composées d'un nombre faible de « **comportements** » différents.

3.2.4.2. Module CONCOURS

Ce module a pour but de réaliser une confrontation entre chaque « **stratégie à tester** » et les « **stratégies environnementales** » ; chaque « **stratégie à tester** » obtient ainsi un score.

Dans un CONCOURS chaque « **stratégie à tester** » affronte chaque « **stratégie environnementale** » pendant le nombre de « **parts** » fixé par l'utilisateur.

A la Figure 3, on voit comment se déroule un tel AFFRONTEMENT entre « **une stratégie à tester** » et « **une stratégie environnementale** ». Après chaque affrontement, la « **stratégie à tester** » obtient un score qui est la somme des scores obtenus à chaque « **part** ».

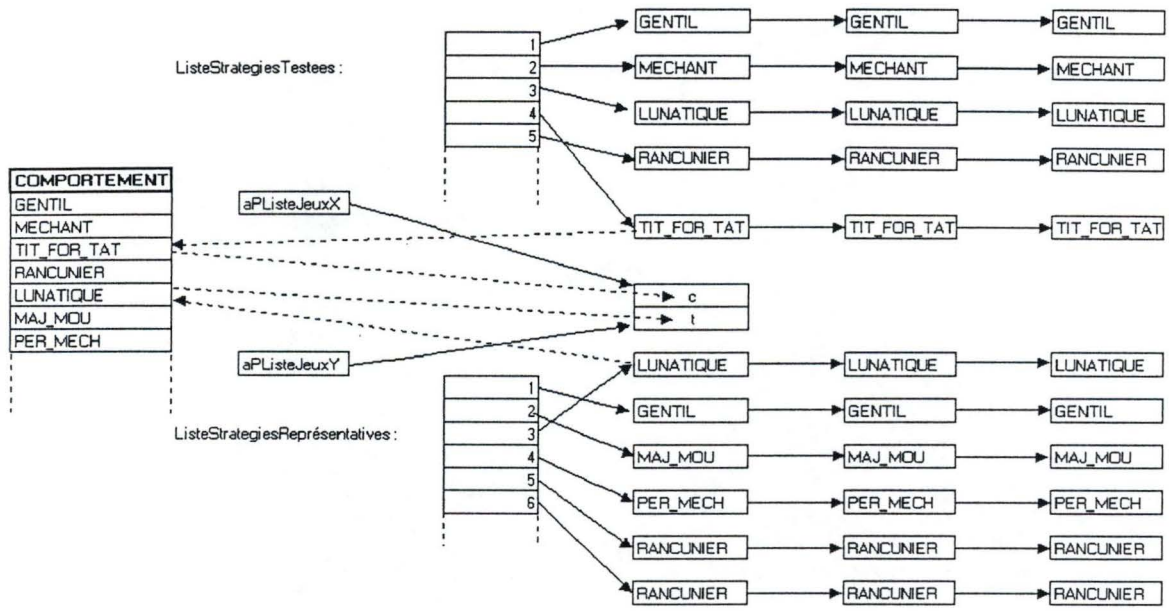

Figure 3 : AFFRONTEMENT entre la quatrième « **stratégie à tester** » et la troisième « **stratégie environnementale** ». Pour des raisons de clarté, nous avons fait comme si l'AFFRONTEMENT entre ces deux « **stratégies** » n'en était qu'à la première « **part** ». Le premier gène de ces deux stratégies est analysé, la fonction correspondant à son comportement est déclenchée pour cette « **part** » et renverra une des trois options possible (renoncer, coopérer, trahir) pour le premier jeu. Les jeux de chaque « **part** » sont conservés pour chaque stratégie pendant tout l'AFFRONTEMENT ; aPListeJeuxX est la liste des jeux de la « **stratégie à tester** » et aPListeJeuxY est la liste des jeux de la « **stratégie représentative** ». Les flèches en trait continu représentent des pointeurs, tandis que les flèches en trait discontinu représentent les appels aux fonctions et le renvoi du jeu joué par ces dernières.

Suite au CONCOURS, chaque « **stratégie à tester** » obtient un score qui est calculé de la façon suivante :

$$\text{ScoreStratégieTestee}[i] = \text{Affrontement}(i, e_1) * \text{Effectif}(e_1) + \ldots + \text{Affrontement}(i, e_n) * \text{Effectif}(e_n)$$

Affrontement (i, e_n) représente le résultat de l'affrontement entre la « **stratégie à tester** » i, et la « **stratégie environnementale** » e_n, Effectif (e_n) est l'effectif de cette dernière.

3.2.4.3. Module SELECTION

La fonction SELECTION permet d'attribuer de nouveaux effectifs aux différentes « **stratégies à tester** » proportionnellement au score qu'elles ont obtenu tout en conservant plus ou moins l'effectif total précédant le CONCOURS, l'effectif total étant la somme des effectifs des « **stratégies à tester** ». Les stratégies ayant obtenu de faibles scores verront donc leur effectif décroître, tandis que celles ayant obtenu de bons scores verront croître leur effectif.

L'algorithme de calcul utilisé afin de définir les nouveaux effectifs est le suivant :

$$\text{EffectifStrategie}[i] = \frac{\text{EffectifTotal} * \text{ScoreStrategie}[i] * \text{EffectifStrategie}[i]}{\text{ScoreTotal}}$$

3.2.4.4. Module CROISEMENT

Un CROISEMENT, correspond au développement d'une génération au moyen de « **reproductions sexuées** » réalisées entre des « **stratégies à tester** »; le nombre de « **reproductions sexuées** » par CROISEMENT étant déterminé par l'utilisateur.

Sélection des stratégies qui effectueront une reproduction sexuée

Une « **reproduction sexuée** » a lieu entre deux « **stratégies à tester** », ces deux stratégies à tester sont tirées au sort ; cependant, notre programme est conçu pour que les stratégies ayant un score plus important suite à la SELECTION aient plus de chance d'être choisies (Figure 4, p 39). En effet, nous réalisons un tableau avec les scores cumulés des différentes stratégies à tester et nous tirons au sort un nombre entre 1 et le score total de toutes les stratégies à tester, la stratégie choisie est celle dont le score cumulé avec les précédentes dépasse le nombre choisi tout en restant le plus petit des scores cumulés. Remarquons également qu'avec ce système, il est possible qu'une stratégie soit choisie plusieurs fois et donc se reproduise plusieurs fois de manière sexuée.

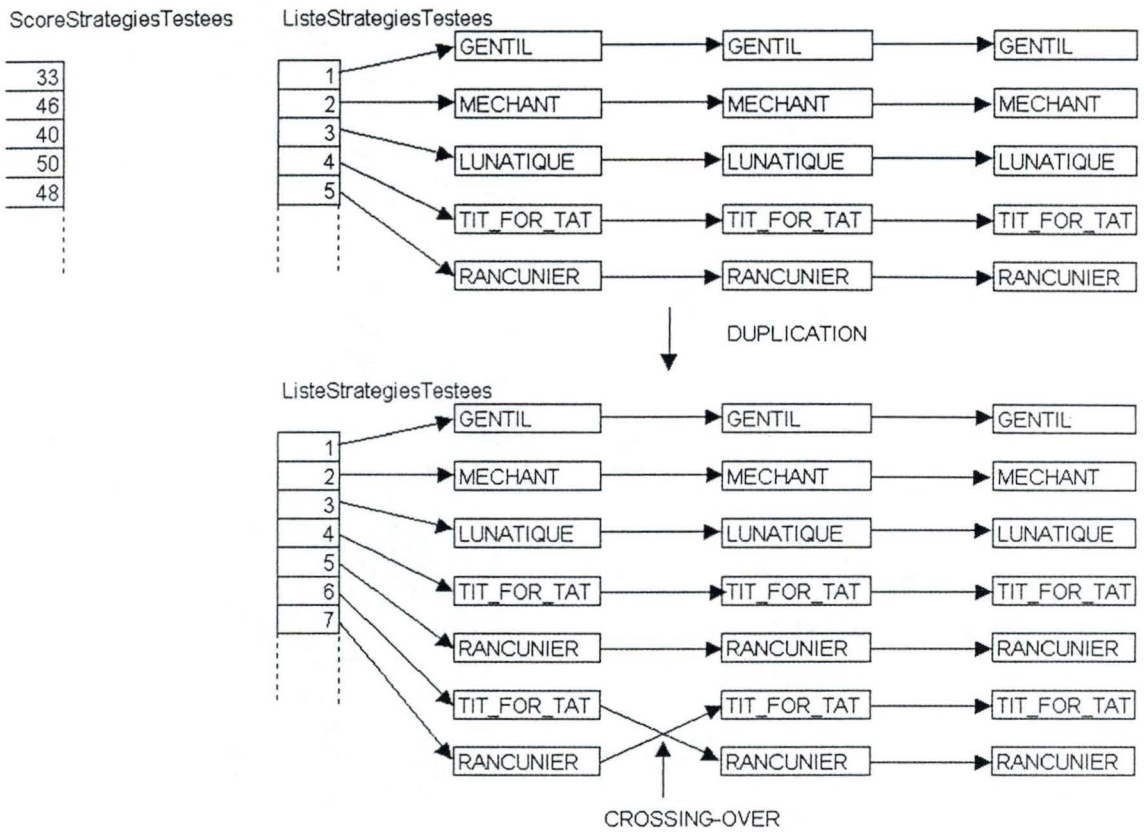

Figure 4 : Duplication de deux « **stratégies à tester** » choisies au hasard (ici TIT_FOR_TAT et RANCUNIER) et ensuite crossing-over entre les deux stratégies résultant de la duplication, on garde ainsi les deux stratégies mères.

3.2.4.5. Module CROSSING_OVER

C'est dans ce module que s'effectue la « **reproduction sexuée** » entre deux « **stratégies à tester** », ces deux stratégies sont d'abord dupliquées et les deux stratégies résultant de cette duplication vont effectuer N échanges de gènes ou crossing-over (Figure 4, p 39); N étant fixé par l'utilisateur. Les nouvelles « **stratégies à tester** » apparaissant de cette manière sont soit ajoutées dans le tableau : ListeStratégiesTestees (déjà été discuté à la section 2.4.2, p 27.

Figure 1, p 30), soit elles remplacent des stratégies à tester ayant obtenu de mauvais scores suite au DEVELOPPEMENT_GENERATIONS et qui ne font pas partie des « **stratégies à tester de départ** ». Ceci afin que toutes les stratégies de départ conservent une chance d'être choisies pour effectuer une reproduction sexuée. Les stratégies remplacées sont supprimées pour améliorer le temps d'exécution du programme. L'effectif en-dessous duquel on décide de remplacer une stratégie est

fixé à 99 ; après le DEVELOPPEMENT_GENERATIONS, certaines stratégies n'atteignent plus cet effectif, alors qu'elles avaient un effectif de 100 chacune avant l'exécution de ce module.

3.2.4.6. Module CONCOURS_ECOLOGIQUE

Ce module a pour but de réaliser une confrontation entre une stratégie obtenue lors de la phase de génération de nouvelles stratégies (« **la meilleure stratégie**», ou la « **stratégie moyenne** ») et les « **stratégies environnementales**» .

Toutes ces stratégies s'affrontant respectivement grâce au module AFFRONTEMENT, recevront un score calculé de la façon suivante :

$$Score(s_i) = Affrontement(s_i,s_1)* Effectif(s_1) + ... + Affrontement(s_i,s_i) * (Effectif(s_i)-1) +...+ Affrontement(s_i, s_n) * Effectif(s_n)$$

avec $s_1,...s_i,...,s_{n-1}$ représentant les n-1 « **stratégies environnementales** » et sn représentant « **la meilleure stratégie** » ou la « **stratégie moyenne** ».

3.3. Présentation des stratégies Implémentées

Nous avons implémenté toutes les stratégies définies dans le Tableau 3, p 13, à l'exception de la stratégie SONDEUR, car cette stratégie n'a plus aucun sens si elle fait l'objet d'un croisement avec une autre. En effet, pour déterminer les coups qu'elle va jouer, cette stratégie a besoin, lors des trois premiers coups, de jouer t, c, c. Or après un « **reproduction sexuée** » avec une autre stratégie, les deux stratégies descendantes n'auront pas forcément joué les trois premiers coups : t, c, c.

En plus des stratégies définies au Tableau 3, p 13, nous avons également implémenté d'autres stratégies que nous présentons dans le tableau suivant :

STRATEGIES SIMPLES	DESCRIPTION		
SEUIL	Si mon résultat moyen est inférieur à 2.5, je trahis, sinon je coopère	d	g
BINAIRE	Je joue périodiquement : coopérer, trahir	i	g
SEUILREN	Si le résultat moyen est inférieur à 2, je renonce, sinon, si le résultat est supérieur à 4, je trahis, sinon je coopère	d	g
MAJ_CINQ	je joue ce que l'autre a joué en majorité dans les cinq derniers coups. Si aucune majorité ne se dégage (possible au début de la confrontation), je coopère	d	g
TESTCONTINUE	si l'adversaire vient de jouer deux fois la même chose, je coopère, sinon je trahis	d	g
PAVLOV	si l'adversaire a joué la même chose que moi au dernier coup, je coopère, sinon je trahis	d	g
PAVLOV_CINQ	si pour les cinq derniers coups, la majorité des coups joués par l'adversaire est la même que la majorité des coups que j'ai joués, je coopère, sinon je trahis	d	g
PROB_MEME	je regarde la probabilité que l'adversaire joue la même chose que ce que j'ai joué la fois précédente. Si lors des coups précédents, lorsque j'ai joué comme le dernier coups, l'adversaire joue la même chose au coup suivant, je coopère, sinon je trahis	d	g
PROB_CONTINUE	je regarde la probabilité que l'adversaire joue la même chose que ce que j'ai joué la fois précédente. Si lors des coups précédents, lorsque j'ai joué comme le dernier coup, l'adversaire joue la même chose au coup suivant, je continue à jouer la même chose, sinon je change	d	g

Tableau 5 : Description de nouvelles stratégies.

1.1. Interface et utilisation du programme

A la Figure 1, on peut voir la première fenêtre du programme ; l'utilisateur doit y indiquer les stratégies qu'il désire tester et l'environnement stratégique (stratégies environnementales) contre lequel elles concourront et qui est fixé pour toute l'expérience.

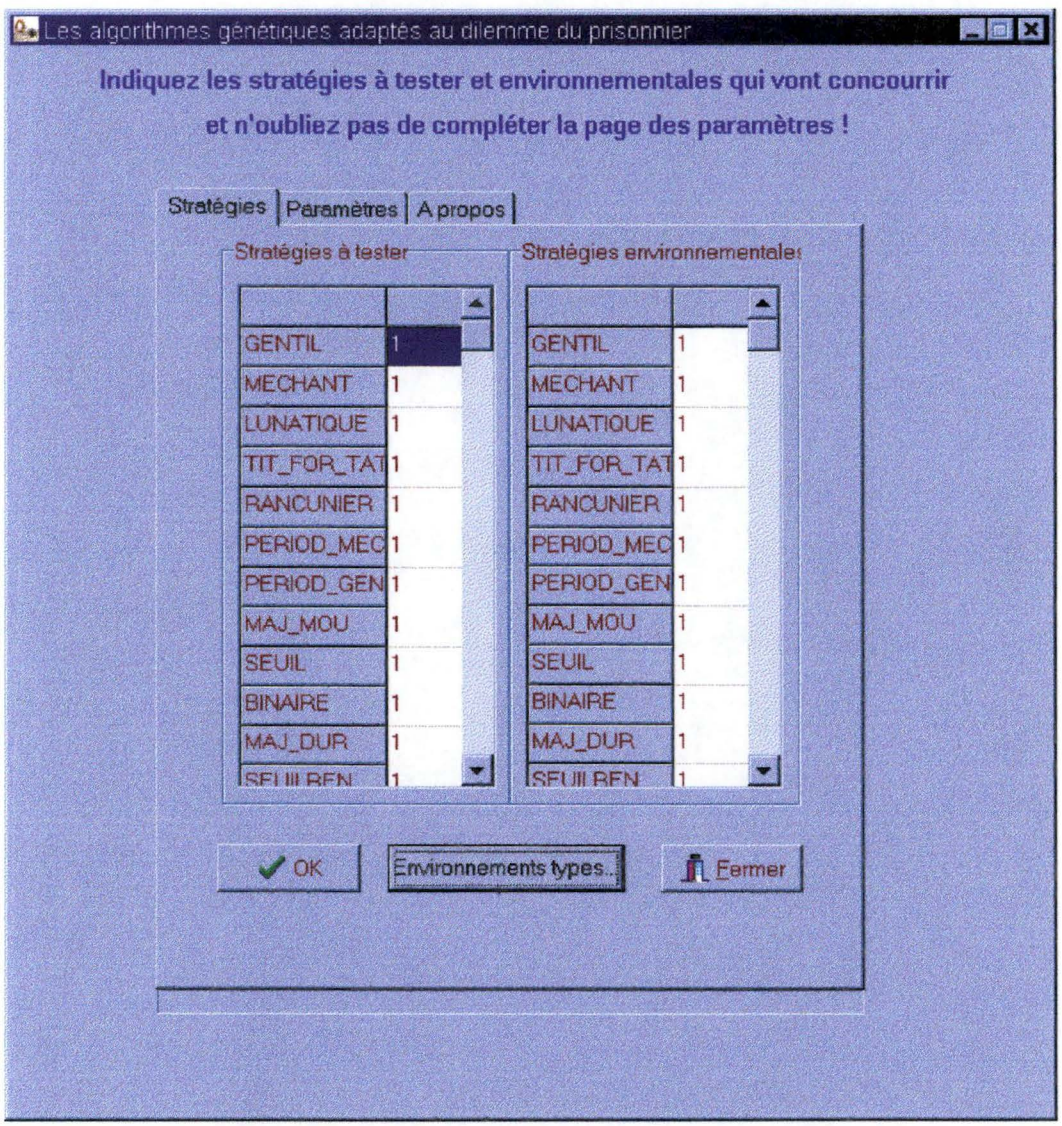

Figure 1 : Fenêtre permettant à l'utilisateur de définir les « **stratégies à tester** » et l'environnement stratégique qu'elles affrontent ; Environnements types... permet de définir automatiquement l'environnement stratégique (voir Figure 2). Dans ce cas, l'utilisateur a choisi toutes les stratégies comme environnement stratégique. Les stratégies à tester sont toutes sélectionnées par défaut. Au bas de la figure 5, une barre de progression permet si les tests durent longtemps d'estimer où en est le programme.

A la Figure 2 on peut voir la fenêtre permettant de sélectionner un environnement type ; nous y avons défini 7 regroupements possibles de stratégies en environnement type. Toutes les stratégies environnementales du programme classées dans l'environnement sélectionné par l'utilisateur se voient accorder un effectif de 1 dans la fenêtre de la Figure 1, p 42, tandis que les autres se voient accorder un effectif de 0.

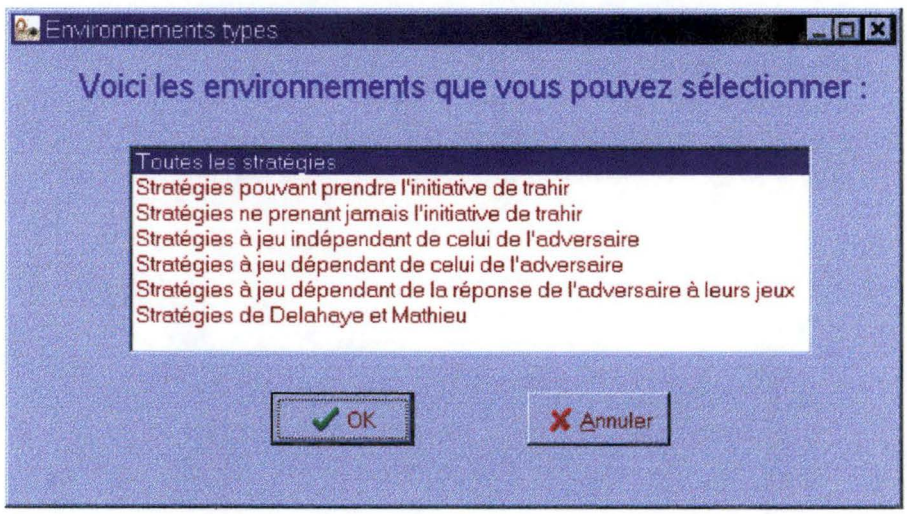

Figure 2 : Sélection automatique d'un environnement type ; un environnement type reprend toutes les stratégies définies dans le programme pouvant être catégorisées dans ce type d'environnement.

A la Figure 3, on peut visualiser la deuxième page de la première fenêtre (Figure 1) que l'on obtient en appuyant sur l'onglet paramètres. C'est dans cette fenêtre que l'utilisateur fixe le nombre de tests, de croisements, de reproductions par croisement, de crossing-over par reproduction et le nombre de coups d'un affrontement entre deux stratégies.

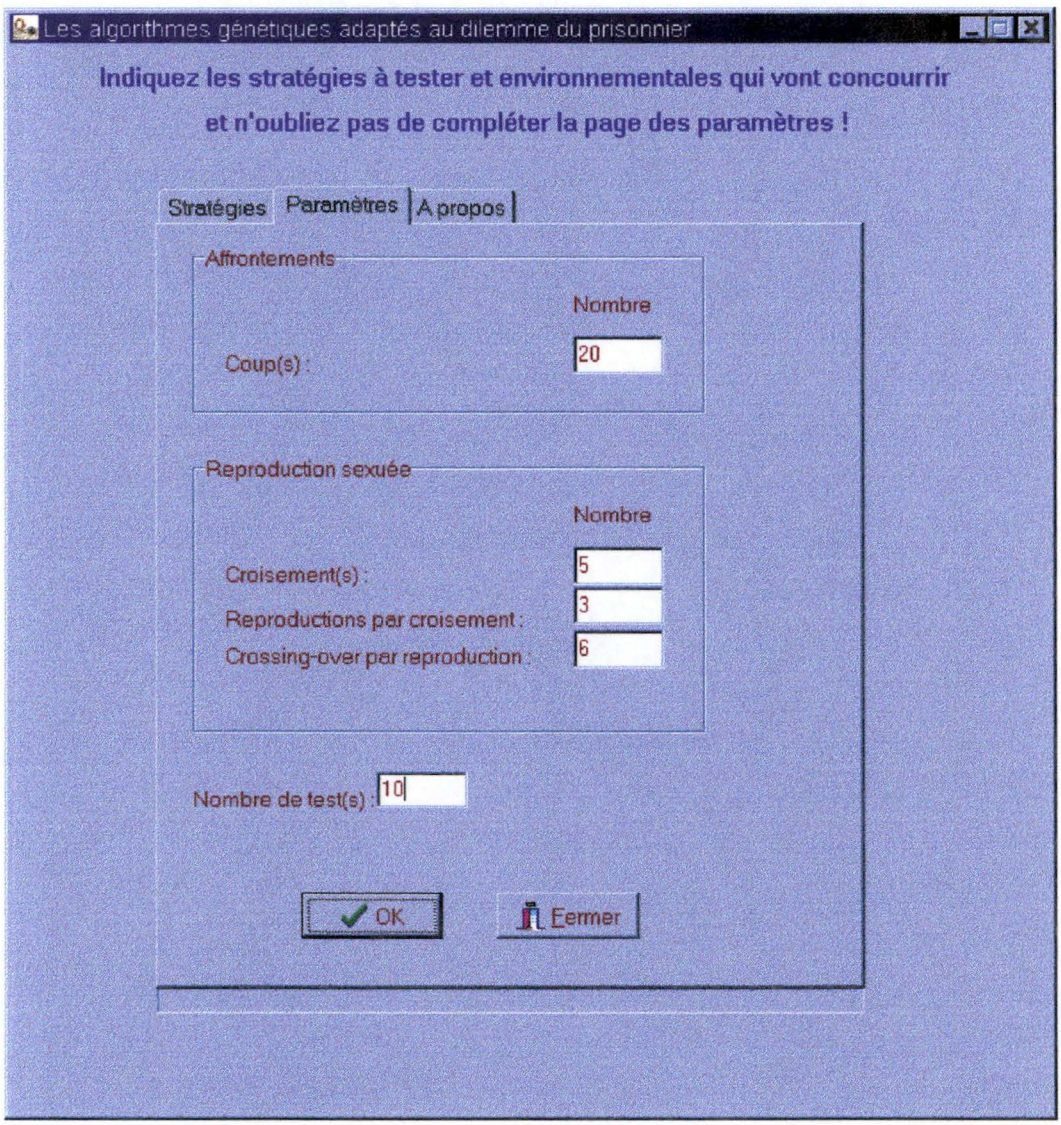

Figure 3 : page de la première fenêtre où l'utilisateur introduit des paramètres. Dans cet exemple, l'utilisateur a décidé qu'un AFFRONTEMENT entre deux stratégies dure 20 coups, c'est-à-dire que les « **stratégies** » seront toutes formées de 20 gènes qui détermineront le jeu joué lors d'un coup. L'utilisateur a également décidé qu'il y aurait 5 croisements, c'est-à-dire que 5 générations seront développées au moyen de reproductions sexuées, ces dernières sont fixées au nombre de 3 par croisement donc 6 stratégies seront choisies, et lors d'une reproduction sexuée entre deux stratégies, il y a 6 crossing-over par reproduction. L'utilisateur a également décidé de répéter 10 fois ce test.

Le résultat de l'évolution apparaît à la Figure 4, sur laquelle on peut voir en moyenne les gènes présents dans la meilleure stratégie de chaque test, ainsi que le gène présent en majorité à une position donnée dans les stratégies gagnantes, nous créons ainsi une nouvelle stratégie que l'on appelle « **stratégie moyenne** ».

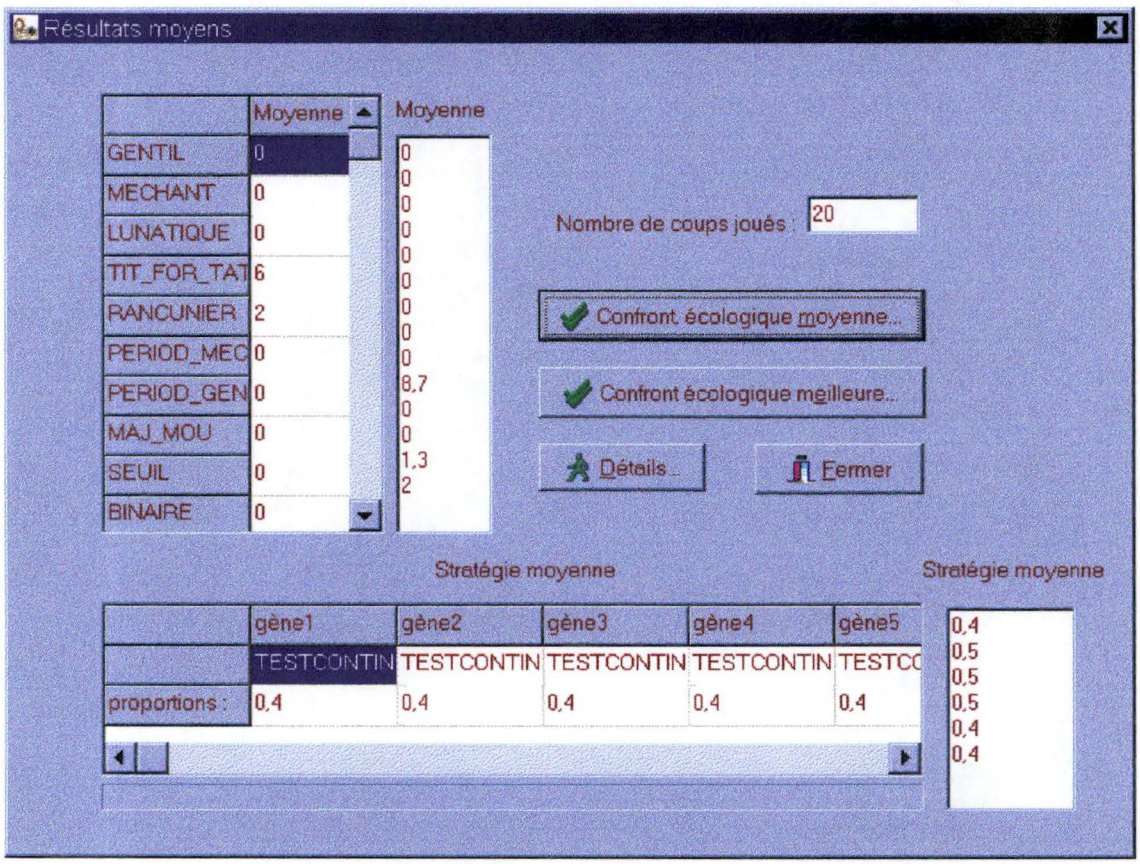

Figure 4 : Résultats de l'évolution dont les caractéristiques ont été fixées par l'utilisateur (Figure 1 et Figure 3). Dans la grille du dessus, on peut voir en moyenne les gènes présents dans la meilleure stratégie à tester de chaque test, dans ce cas en moyenne on a 6 gènes TIT_FOR_TAT sur 20 gènes (coups). Dans la grille du dessous, on peut voir le gène présent en majorité à une position donnée dans les stratégies gagnantes, dans ce cas-ci c'est TESTCONTINUE avec une proportion de 0,4 aux positions 1 à 5, ce qui veut dire que sur les dix tests, à ces positions le gène TESTCONTINUE est présent quatre fois sur dix. Détail... permet de visualiser la « **meilleure stratégie** » de chaque test. Confront. écologique moyenne et Confront. écologique meilleure, permettent de réaliser une confrontation écologique avec comme stratégies respectivement la « **stratégie moyenne** », ou la « **meilleure stratégie de l'expérience** », et les stratégies environnementales choisies en Figure 1.

La Figure 5 présente le détail des meilleures stratégies de chaque test.

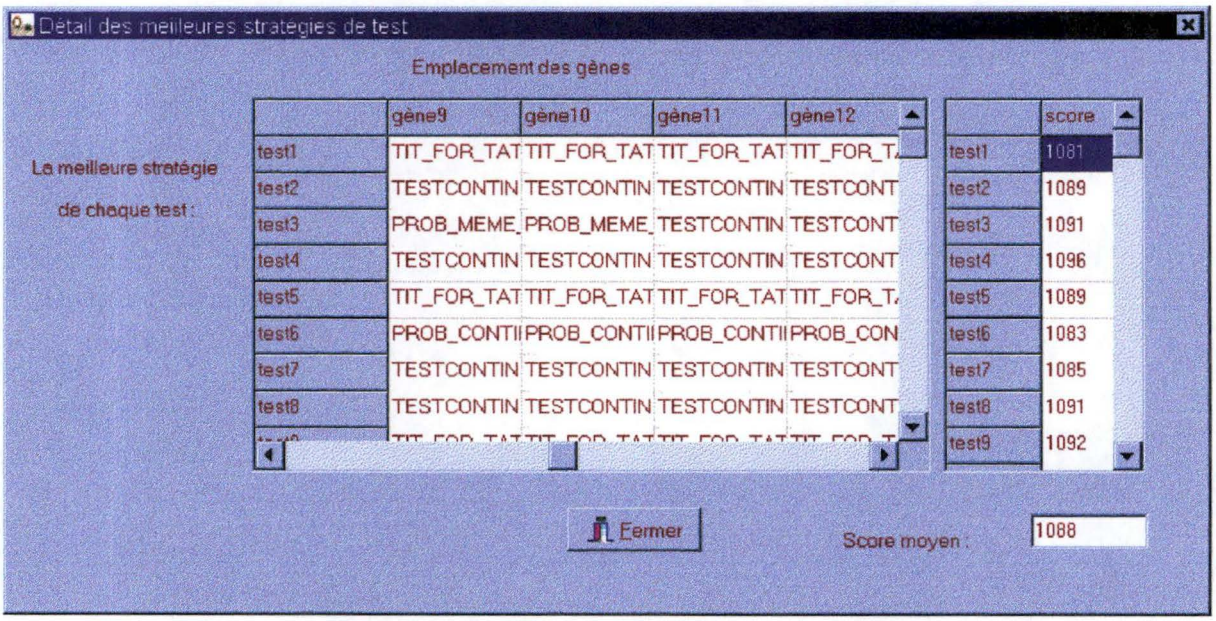

Figure 5 : Détail des gènes de la meilleure stratégie de chaque test (grille de gauche) avec son score (grille de droite).

La Figure 6 présente l'évolution des effectifs des stratégies lors de la confrontation écologique réalisée.

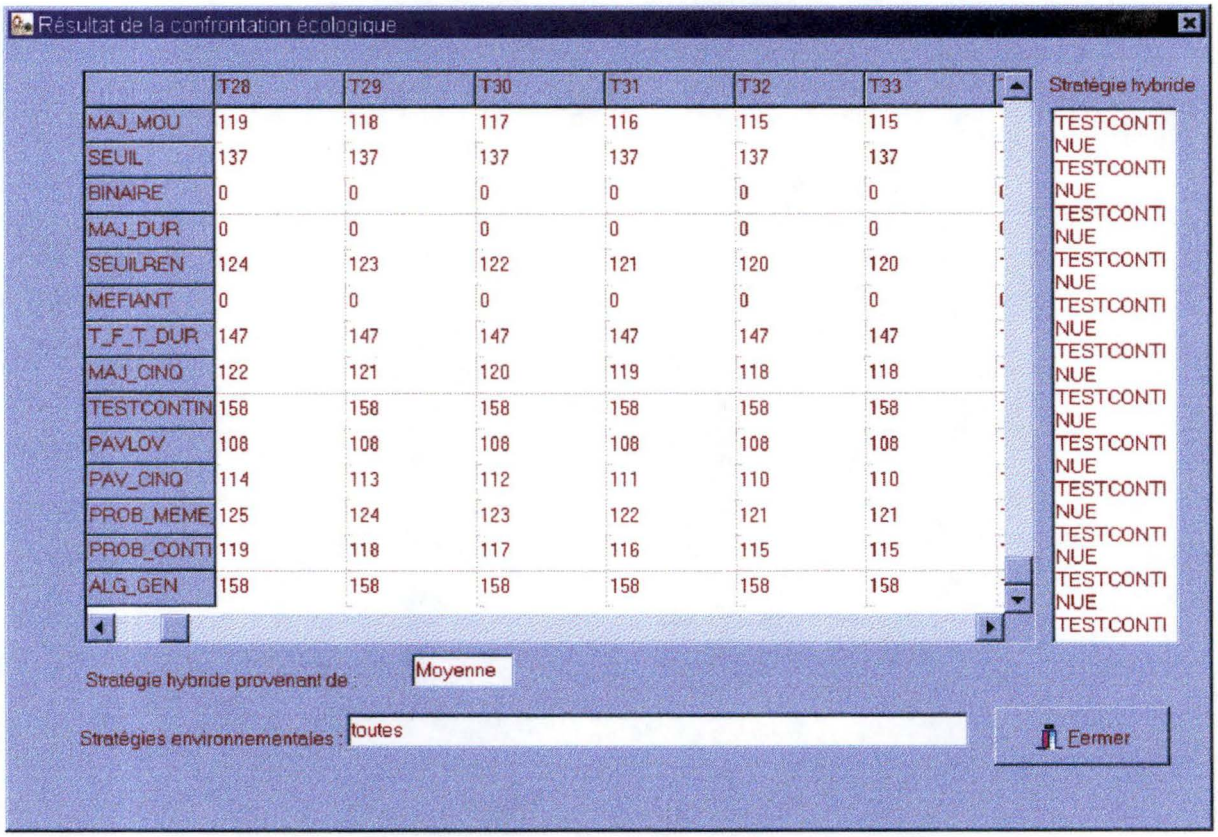

Figure 6 : Evolution de l'effectif des stratégies lors de la confrontation écologique. Au temps T0, chacune possédait un effectif de 100 individus. La stratégie nommée ALG_GEN représente la stratégie issue de la phase de génération de nouvelles stratégies, dans ce cas-ci, l'utilisateur a réalisé une confrontation écologique avec « **la stratégie moyenne** ».

Chapitre IV : Présentation des expériences et des résultats

Nous avons procédé à plusieurs types d'expérience ; ainsi, dans les premières expériences, nous avons essayé de déterminer approximativement quelles valeurs de nos paramètres (Figure 7, p 44) nous permettent de réaliser des expériences rapidement avec l'assurance de trouver des stratégies obtenant de très bons scores dans un environnement donné (section 4.1, p 48). Ensuite, nous avons essayé de voir dans quelles proportions les meilleures stratégies obtenues, trahissaient ou coopéraient, et ceci, en relation avec l'environnement stratégique dans lequel elles sont apparues(section 4.2, p 59). Dans le dernier type d'expérience, nous avons essayé de "faire naître" dans divers environnements des stratégies « **robustes** » (section 4.3, p 71 et section 4.4, p 83).

4.1 L'influence des paramètres sur les performances du programme

Nous avons étudié l'influence des paramètres sur les performances du programme à l'aide de quelques expériences. Lors d'une expérience ou si on préfère d'une exécution du programme, on réalise un certain nombre de tests.

L'augmentation du nombre de tests par expérience permet en fait d'accroître la précision des moyennes (la moyenne des scores des meilleures stratégies des différents tests (Figure 9, p 46), la « **stratégie moyenne** »). Il est important d'essayer de voir l'effet des paramètres sur le temps d'exécution et sur la "qualité" des stratégies obtenues, afin de réaliser en un minimum de temps des expériences nous permettant de trouver de bonnes stratégies.

4.1.1. Les stratégies de départ sont GENTIL et MECHANT

4.1.1.1. Le nombre de croisements et de reproductions par croisement

Pour étudier plus précisément l'effet des croisements et du nombre de reproductions, nous avons réalisé des expériences où nous faisions varier ces deux paramètres, tout en gardant leur produit constant de manière à explorer un même

nombre de solutions à chaque fois ; ainsi, en réalisant 200 croisements et une reproduction par croisement, comme à chaque reproduction 2 stratégies apparaissent, nous explorons 400 solutions. Si nous n'effectuons que 100 croisements, nous réalisons donc 2 reproductions par croisement afin d'explorer 400 solutions comme précédemment.

	score	temps (sec)
200 croisements avec une reproduction	2773	80
100 croisements avec 2 reproductions	2774	65
50 croisements avec 4 reproductions	2775	44
25 croisements avec 8 reproductions	2770	17
10 croisements et 20 reproductions	2757	7
5 croisements et 40 reproductions	2747	5
2 croisement et 100 reproductions	2700	5

Tableau 6 : variation du score moyen des meilleures stratégies et du temps d'exécution du programme pour des expériences où l'on réalise des affrontements de 50 parts, 1 crossing-over par reproduction, 50 tests et où varient uniquement le nombre de croisements et de reproduction(s) par croisement. Les stratégies à tester de départ étaient GENTIL et MECHANT que nous avons confrontées à l'environnement stratégique composé de toutes les stratégies du programme. Le score est le score moyen des meilleures stratégies des 50 tests d'une expérience. Ces expériences ont été réalisées sous Windows 98, sur un PC (Pentium II, 333 Mhz, 64 Mo de RAM).

Sur le Tableau 6 , on peut voir que l'expérience pour laquelle sont réalisés 8 reproductions par croisement et 25 croisements permet d'explorer 400 nouvelles stratégies en un temps raisonnable et également de découvrir d'aussi bonnes stratégies en moyenne que dans les expériences où le nombre de croisements est plus important. Par contre, en-dessous de 25 croisements, les scores moyens obtenus semblent nettement plus faibles.

On peut également voir que le temps d'exécution peut s'allonger considérablement lorsqu'on augmente le nombre de croisements ; à la fin du test où l'on effectue 200 croisements, 360 stratégies sont apparues, tandis que 323 sont apparues lorsqu'on a réalisé le même test avec 100 croisements et 2 reproductions par croisement, et 27 stratégies seulement ont été créées dans l'expérience où l'on réalisait 25 croisements et 8 reproductions par croisement. Il est logique que le temps d'exécution du programme s'allonge quand le nombre de « **stratégies à tester** » augmente ; car beaucoup de boucles s'exécutent jusqu'à ce nombre.

On peut donc penser que les stratégies apparaissant quand le nombre de croisements diminue et que le nombre de reproductions par croisement augmente, sont

le plus souvent des stratégies obtenant un faible score et qui ont donc plus de chance d'être supprimées dans le module CROSSING_OVER (section 3.2.4.5, p 39).

Nous supposons que dans l'expérience où l'on réalise 200 croisements et 1 reproduction par croisement, la plupart des stratégies sont issues directement ou indirectement d'une stratégie créée dans les premiers croisements qui, vu son score important par rapport à GENTIL et MECHANT s'est peut-être reproduite avec elle-même et aurait donc donné naissance à deux nouvelles stratégies identiques à elle-même. Une stratégie pourrait ainsi rapidement se trouver en plusieurs exemplaires, et si ces exemplaires obtiennent de bons scores, il seront rarement supprimés. Par contre, lorsque nous augmentons le nombre de reproductions par croisement, une stratégie même si elle obtient de très bons scores a beaucoup moins de chances d'être croisée à elle-même et donc de produire d'autres stratégies qui lui sont identiques et qui obtiendraient également de bons scores ; les nouvelles stratégies produites auraient donc plus de chance d'être des mauvaises stratégies et donc d'être éliminées, que dans l'expérience avec 1 reproduction par croisement. Si cette hypothèse est correcte, dans le cas, où l'on réalise peu de reproduction(s) par croisement, il sera difficile de faire naître d'autres bonnes stratégies fort différentes de celle qui s'est reproduite avec elle-même.

Pour cette raison, nous avons optimisé le programme et n'avons plus permis qu'une stratégie se reproduise avec elle-même. Cependant, il reste fort probable que les meilleures stratégies qui se reproduisent se ressemblent encore très fort. Dans ce qui suit, nous parlons de la mutation qui est une technique susceptible de réduire l'importance de ces problèmes.

Lors d'une mutation, un gène (un comportement) est remplacé par un autre gène (un comportement que nous avons programmé). Dans nos expériences, nous n'avons pas jugé utile de réaliser des mutations, car beaucoup d'auteurs suggèrent un taux de mutation très faible autour de 1 pour 1000 bits (D. Glover, 1988) , donc 1 pour 1000 gènes dans nos expériences, or, nous n'avons jamais dépassé les 200 gènes (coups). Néanmoins, il serait probablement intéressant dans l'avenir d'ajouter un module « MUTATION » à notre programme, afin de tester l'effet de ces mutations sur les scores moyens des meilleures stratégies de l'expérience. Ce module pourrait ainsi être intéressant quand la longueur des affrontements est très importante, il permettrait

d'obtenir plus facilement qu'avec la technique des crossing-over un gène isolé dans une séquence de gènes de même comportement. En effet, par la technique des crossing-over, nous obtenons souvent des séquences assez longues de gènes adoptant un seul comportement puisque lors de nos expériences nous choisissons un nombre de crossing-over assez faible (voir section 4.1.1.3, p 53).

4.1.1.2. Le nombre de stratégies explorées

Lorsque l'on augmente uniquement le nombre de croisements, on explore plus de possibilités et la moyenne des scores des « **meilleures stratégies des tests** » s'élève jusqu'à peu près 2791 (Figure 11) , ensuite, elle a tendance à se stabiliser au-dessus de 1600 stratégies explorées. On peut également remarquer que les scores des meilleures stratégies testées augmentent difficilement au-delà de 800 stratégies explorées, donc de 50 croisements et 8 reproductions par croisement.

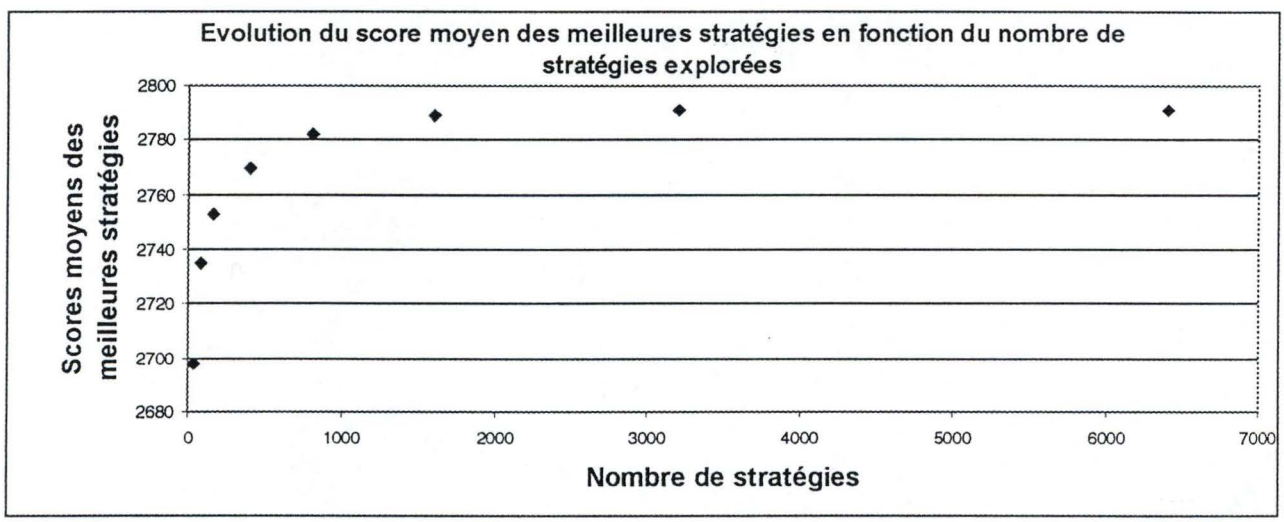

Figure 11 : score moyen des meilleures stratégies en fonction du nombre de stratégies explorées. Les paramètres constants de ces 8 expériences sont : 50 coups, 8 reproductions par croisement, 1 crossing-over et 50 tests. Les stratégies à tester de départ sont uniquement des GENTILS et des MECHANTS et l'environnement qu'elles affrontent est composé de toutes les stratégies du programme. On augmente le nombre de stratégies explorées en augmentant le nombre de croisements.

On peut également remarquer que pour chaque expérience, le score moyen des « **meilleures stratégies des tests** » est nettement meilleur que le score obtenu par les « **stratégies simples** » GENTIL et MECHANT dans les mêmes conditions et qui sont respectivement de 2545 et de 2189 (Tableau 7, p 52).

	s ores		
TIT_FOR_TAT	2713	g	d
TESTCONTINUE	2709	g	d
RANCUNIER	2691	g	d
PROB_MEME	2681	g	d
PROB_CONTINUE	2672	g	d
PAV_CINQ	2668	g	d
T_F_T_DUR	2661	g	d
MAJ_CINQ	2661	g	d
SEUIL	2654	g	d
MAJ_MOU	2649	g	d
SEUILREN	2641	g	d
GENTIL	2545	g	i
PERIOD_GENT	2538	m	i
PAVLOV	2538	g	d
BINAIRE	2221	m	i
MECHANT	2189	m	i
MAJ_DUR	2177	m	d
LUNATIQUE	2131	m	i
MEFIANT	2105	m	d
PERIOD_MECH	1929	m	i

Tableau 7 : score moyen de chaque stratégie dans une expérience où elle réalise une confrontation généralisée avec toutes les stratégies du programme, 50 tests sont effectués, l'affrontement dure 50 parts, et nous n'effectuons pas de croisement. Dans la troisième colonne : un « g » signifie que la stratégie est « **gentille** » et un « m » signifie qu'elle est « **méchante** ». Dans la quatrième colonne : un « i » signifie qu'il s'agit d'une stratégie à jeux indépendants de celui de son adversaire, tandis qu'un « d » signifie qu'il s'agit d'une stratégie à jeux dépendant de celui de l'adversaire.

Sur ce tableau TIT_FOR_TAT obtient le plus beau score, ce qui est en accord avec les conclusions de J. P. Delahaye & Ph. Mathieu (1992) et R. Axelrod (1984) concernant la supériorité de TIT_FOR_TAT. Le score de TIT_FOR_TAT est cependant bien en-dessous de la limite de score moyen (2791) (Figure 11, p 51) que nous avons obtenu en mélangeant les gènes de GENTIL et de MECHANT.

4.1.1.3. Le taux de crossing-over

Il semble qu'augmenter la proportion de crossing-over par rapport au nombre de parts, ne permette pas d'obtenir de meilleurs résultats, du moins pour des affrontements ne durant pas plus de 100 parts et dans les expériences que nous avons réalisées avec des « **stratégies à tester de départ** » qui étaient uniquement des GENTILS et MECHANTS. Beaucoup d'auteurs proposent des taux de crossing-over de l'ordre de 60 à 70 % (D. Glover, 1988), nous avons testé ces taux et pu constater qu'ils ne donnaient pas de meilleurs résultats qu'en ne réalisant qu'un crossing-over dans toute la stratégie (chromosome) (Tableau 8).

		rossi g-o er	7 rossi g-o ers ar o s
	o s	583	573
	o s	1135	1053
3	o s	1680	1588
	o s	2225	2059
	o s	2770	2576

Tableau 8 : scores moyens des meilleures stratégies pour chaque expérience en réalisant 50 tests 25 croisements, 8 reproductions par croisement et en faisant varier le nombre de coups. Les « **stratégies à tester de départ** » sont uniquement des GENTILS et des MECHANTS et l'environnement qu'elles affrontent est composé de toutes les stratégies du programme. On voit que les meilleures stratégies sont trouvées quand on ne réalise qu'un crossing-over par reproduction.

4.1.1.4. Autres remarques concernant les paramètres

Les meilleures stratégies trouvées en réalisant un nombre impair de crossing-over, obtiennent en général de bien meilleurs scores que les meilleures stratégies obtenues en réalisant un nombre pair de crossing-over (Figure 12, p 54), lorsque le nombre de coups par affrontement est de 50, nous avons réalisé d'autres expériences avec des affrontements de 100 et 200 coups (**Annexe 7**, p 103, **Annexe 8**, p 104), et il semble que le même phénomène s'observe. Le meilleur score moyen étant réalisé lorsque l'on n'effectue qu'un crossing-over par reproduction. Nous ne pouvons pas donner d'explications à ce phénomène, mais nous n'excluons pas qu'il soit dû à une erreur dans notre programme, malgré toutes les vérifications que nous avons effectuées.

Ceci pourrait également s'expliquer par le fait que l'augmentation du nombre de crossing-over introduit une certaine incohérence dans la stratégie du fait du changement fréquent de comportement. Comme l'ont déjà souligné Delahaye et Mathieu (1993), il semble préférable d'adopter un comportement clair.

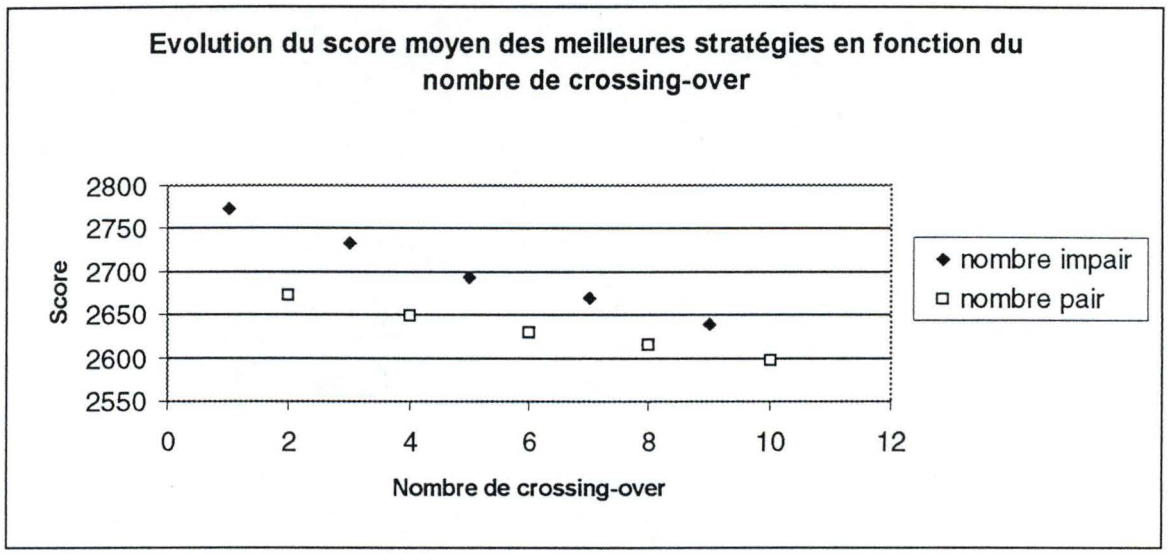

Figure 12 : effet du nombre de crossing-over sur le score moyen des meilleures stratégies obtenues. Les valeurs de paramètres constantes sont : 50 tests, 50 coups par affrontement, 8 reproductions par croisement et 25 croisements.

4.1.2. Tests des paramètres avec toutes les stratégies

Nous avons réalisé le même genre d'expérience que précédemment pour essayer de fixer les valeurs de paramètres les plus optimales possibles, mais cette fois en utilisant comme stratégies à tester de départ toutes les stratégies de notre programme.

4.1.2.1. Le nombre de croisements et de reproduction(s) par croisement

Comme dans l'expérience de la section 4.1.1.1, p 48, il semble qu'il vaille mieux faire plus de croisements et donc moins de reproductions par croisement si l'on explore un même nombre de stratégies (Tableau 9, p 55).

	s ore	tem s se
roiseme ts a e e re rod tio	2844	90
roiseme ts a e re rod tio s	2840	63
roiseme ts a e re rod tio s	2824	32
roiseme ts a e re rod tio s	2792	25
roiseme ts et re rod tio s	2776	11
roiseme ts et re rod tio s	2767	7
roiseme t et re rod tio s	2743	5

Tableau 9 : variation du score moyen des meilleures stratégies et du temps d'exécution du programme pour des expériences où l'on réalise des affrontements de 50 parts, 1 crossing-over par reproduction, 50 tests et où varient uniquement le nombre de croisements et de reproduction(s) par croisement. Les stratégies à tester de départ sont toutes les stratégies du programme que nous avons confrontées à l'environnement stratégique composé de toutes les stratégies du programme. Le score est le score moyen des meilleures stratégies des 50 tests de l'expérience.

Le nombre de reproduction(s) par croisement

Il semble que le nombre de reproduction par croisement qui soit optimal se situe entre 5 et 10. Cependant, ces expériences n'ont été réalisées que pour 3 nombres de croisements différents (10, 25, 50) et avec les autres paramètres fixés (

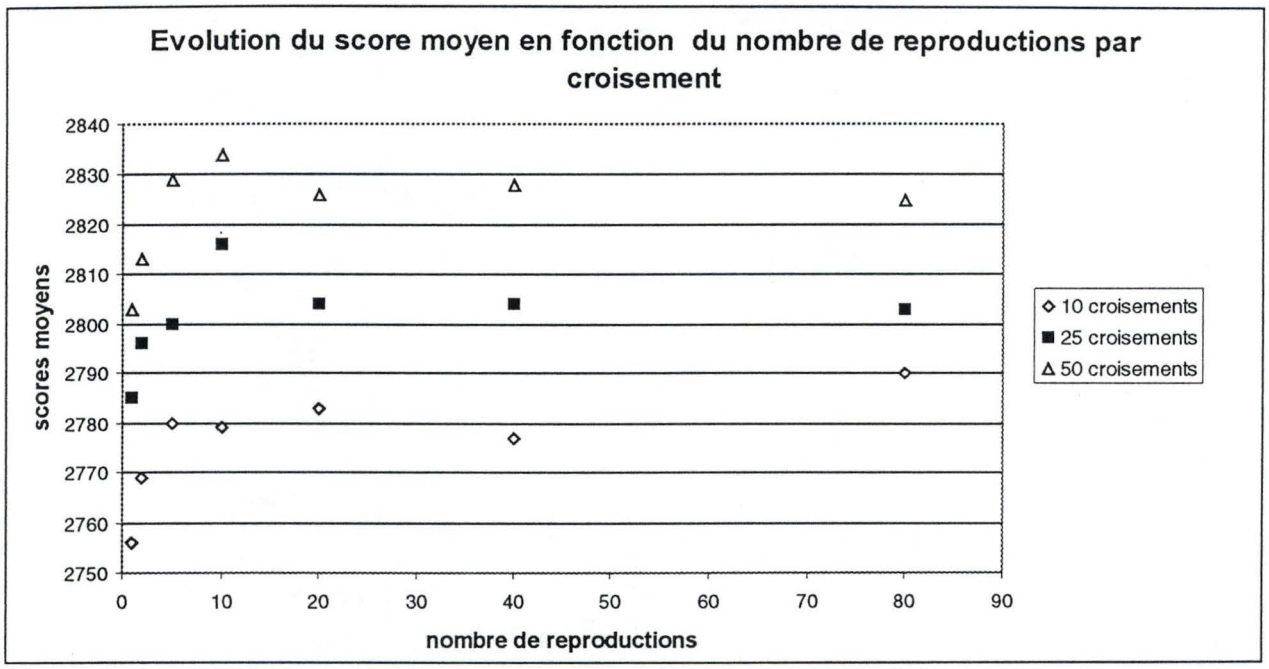

Figure 13).

Figure 13 : Evolution du score moyen en fonction du nombre de reproduction par croisement, dans les cas où on effectue 10, 25 et 50 croisements, en effectuant chaque fois 50 coups par affrontement, 1 crossing-over par reproduction et 50 tests.

4.1.2.2. Le nombre de stratégies explorées

Tout comme à la section 4.1.1.2 (p 51), il existe une limite supérieure au score moyen que l'on peut obtenir en augmentant le nombre de stratégies explorées si l'on utilise comme valeurs constantes de paramètre (50 coups, 8 reproductions par croisement, 1 crossing-over par reproduction et 50 tests) et que l'on fait varier le nombre de croisements (Figure 14, p 57).

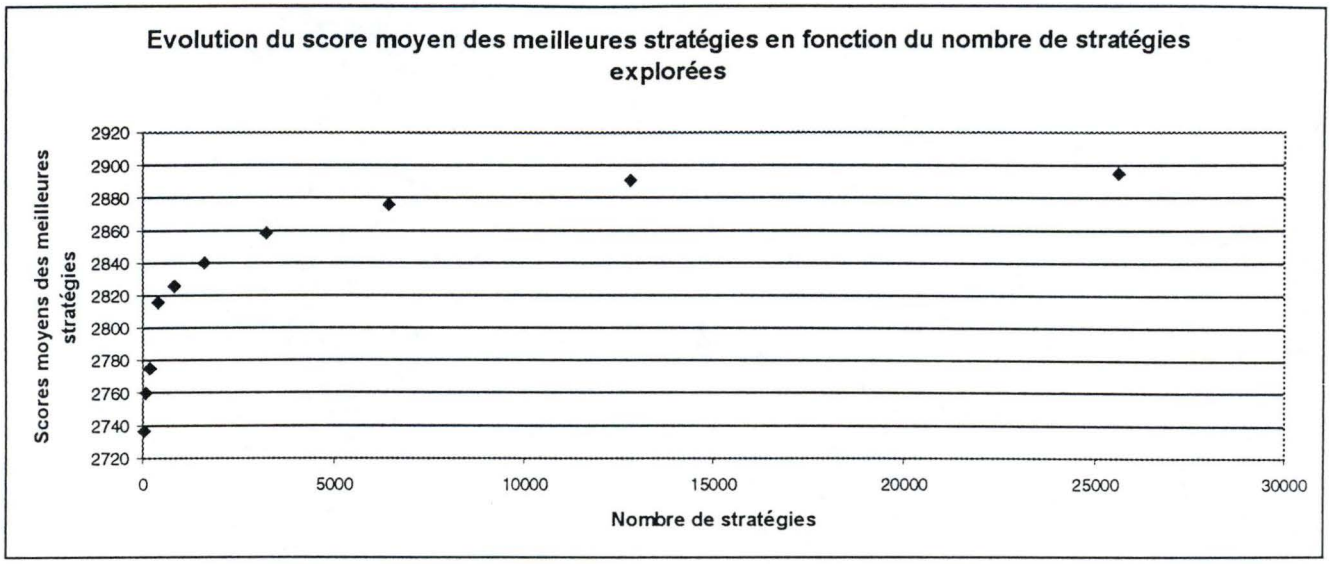

Figure 14 : score moyen des meilleures stratégies en fonction du nombre de stratégies explorées. Les paramètres constants de ces 8 expériences sont : 50 coups, 8 reproductions par croisement, 1 crossing-over et 50 tests. Les stratégies à tester de départ sont toutes les stratégies du programme et l'environnement qu'elles affrontent est composé de toutes les stratégies du programme. On augmente le nombre de stratégies explorées en augmentant le nombre de croisements.

On peut voir que le score moyen des meilleures stratégies se stabilise autour de 2891 points lorsqu'on a exploré 12800 stratégies. Les algorithmes génétiques nous permettent donc d'obtenir des stratégies dont les scores sont près de 200 points supérieurs à celui de TIT_FOR_TAT (la stratégie programmée ayant obtenu le plus beau score (Tableau 7, p 52)). Ces nouvelles stratégies peuvent donc faire en moyenne presque 4 points de plus par coup que TIT_FOR_TAT dans la « **confrontation généralisée** » mettant en jeu les 20 stratégies de notre programme donc 0.5 points en plus par coup et par stratégie affrontée.

On peut également remarquer que ce score moyen de plus ou moins 2895 points est également bien meilleur que celui obtenu lorsque les « **stratégies à tester de départ** » étaient uniquement GENTIL et MECHANT (section 4.1.1.2, p 51) et où les scores moyens des stratégies atteignaient au maximum 2791 points. Cette observation nous permet de penser que pour obtenir plus de points, il vaut mieux avoir des gènes capables de jouer en fonction du jeu précédent de l'adversaire, ce dont n'est pas capable un gène MECHANT ou GENTIL.

4.1.2.3. Le nombre de crossing-over

Des expériences en faisant varier le taux de crossing-over dans le chromosome ou stratégie semblent à nouveau montrer qu'il ne sert à rien de réaliser plus d'un crossing-over pour des affrontements de 50 coups ; et pour des affrontements de 100 coups, nous observons nettement qu'il vaut mieux ne réaliser qu'un crossing-over par reproduction comme c'était le cas à la section 4.1.1.4 (p 53). Il n'est plus si clair qu'il vaille mieux réaliser un nombre impair de crossing-over par reproduction.

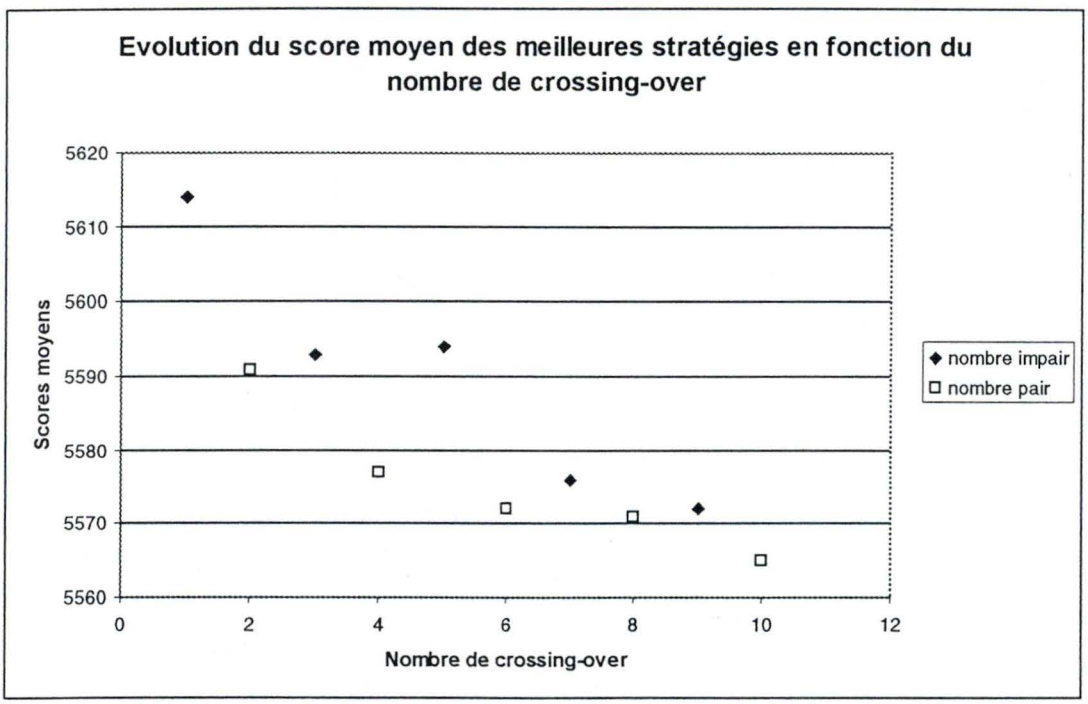

Figure 15: effet du nombre de crossing-over sur le score moyen des meilleures stratégies obtenues. Les valeurs de paramètres constantes sont : 50 tests, 100 coups par affrontement, 8 reproductions par croisement et 25 croisements.

4.1.3. Conclusions

Ces expériences nous laissent penser que des paramètres adéquats, tant que les expériences ne dépassent pas 200 coups par affrontement, sont par exemple : 100 croisements, 8 reproductions par croisement et 1 crossing-over par reproduction. En effet, quand on augmente le nombre de croisements et de reproductions par croisement, le temps d'exécution du programme augmente sensiblement et les « **meilleures stratégies de test** » n'obtiennent pas des scores nettement supérieurs.

Quand à l'augmentation du nombre de crossing-over, elle semble provoquer une diminution des scores des meilleures stratégies.

4.2. Analyse de nos environnements stratégiques

Pour analyser dans quelles proportions les stratégies obtenues dans un environnement donné jouaient la coopération ou la trahison, nous n'avons utilisé comme « **stratégies à tester** » que les stratégies MECHANT et GENTIL, puisqu'à chaque coup, ces stratégies jouent respectivement trahir et coopérer.

4.2.1. Définition de nos environnements stratégiques

Pour plus de facilité, nous avons repris les différents environnements stratégiques avec les stratégies qui en font partie dans le Tableau 10; pour rappel, les définitions des stratégies utilisées dans l'expérience de J. P. Delahaye & Ph. Mathieu (1992) se trouvent au Tableau 3, (p 13)et les autres au Tableau 5, (p 41).

Toutes	pouvant prendre l'initiative de trahir	ne prenant jamais l'initiative de trahir	à jeux indépendants de celui de l'adversaire	à jeux dépendants de celui de l'adversaire	à jeux dépendants de la réponse de l'adversaire à son jeu
GENTIL		GENTIL	GENTIL		
MECHANT	MECHANT		MECHANT		
LUNATIQUE	LUNATIQUE		LUNATIQUE		
TIT_FOR_TAT		TIT_FOR_TAT		TIT_FOR_TAT	
RANCUNIER		RANCUNIER		RANCUNIER	
PERIOD_MECH	PERIOD_MECH		PERIOD_MECH		
PERIOD_GENT	PERIOD_GENT		PERIOD_GENT		
MAJ_MOU		MAJ_MOU		MAJ_MOU	
SEUIL		SEUIL		SEUIL	
BINAIRE	BINAIRE		BINAIRE		
MAJ_DUR	MAJ_DUR			MAJ_DUR	
SEUILREN		SEUILREN		SEUILREN	
MEFIANT	MEFIANT			MEFIANT	
T_F_T_DUR		T_F_T_DUR		T_F_T_DUR	
MAJ_CINQ		MAJ_CINQ		MAJ_CINQ	
TESTCONTINUE		TESTCONTINUE		TESTCONTINUE	
PAVLOV		PAVLOV		PAVLOV	PAVLOV
PAV_CINQ		PAV_CINQ		PAV_CINQ	PAV_CINQ
PROB_MEME		PROB_MEME		PROB_MEME	PROB_MEME
PROB_CONTINUE		PROB_CONTINUE		PROB_CONTINUE	PROB_CONTINUE

Tableau 10 : composition des différents environnements stratégiques.

D'autres environnements stratégiques pourraient facilement être définis à partir d'autres caractéristiques rencontrées chez les stratégies. Dans nos expériences, nous nous sommes limités aux 6 environnements de ce tableau.

L'environnement stratégique de la première colonne est composé de toutes les « **stratégies simples** » que nous avons encodées dans le programme.

L'environnement de la deuxième colonne comprend les « **stratégies simples** » qui sont susceptibles de trahir en premier dans un affrontement, mais qui ne trahissent pas nécessairement en premier, rappelons par exemple que la « **stratégie simple** » PERIODIQUE_MECHANT ne trahit qu'à la troisième part de l'affrontement.

L'environnement de la troisième colonne est celui des « **stratégies simples** » qui ne trahissent jamais en premier, si une de celles-ci trahit, c'est donc que la stratégie qu'elle affronte l'a nécessairement trahie auparavant. Cette trahison n'est en quelque sorte qu'une punition.

L'environnement de la quatrième colonne reprend les « **stratégies simples** » à jeux indépendants de ceux de l'adversaire ; en ce sens que ce qu'elles joueront lors d'une part ne sera pas influencé par ce qu'a joué leur adversaire lors des parts précédentes ; ainsi la stratégie LUNATIQUE jouera toujours au hasard.

Dans l'environnement des « **stratégies simples** » à jeux dépendants de celui de l'adversaire, nous n'avons que des stratégies qui réagissent aux trahisons de l'adversaire en le trahissant, et qui réagissent à ses coopérations en coopérant. Une stratégie qui trahirait en réaction à une ou de(s) coopération(s) pourrait néanmoins faire partie de cet environnement. Si des stratégies de ce type existaient dans notre programme, on aurait pu scinder l'environnement des stratégies à jeux dépendants de celui de l'adversaire en plusieurs sous-environnements.

L'environnement des « **stratégies simples** » à jeux dépendants de la réponse de l'adversaire à leurs jeux comprend des stratégies qui tiennent compte à la fois du ou des jeux joué(s) par leur adversaire lors de part(s) précédente(s), mais également du ou des jeux qu'elles ont joués lors des parts antérieures aux jeux de l'adversaire dont elles tiennent compte.

4.2.2. Evolution de la proportion des trahisons et coopérations en fonction du nombre de coups joués

Afin d'explorer un grand nombre de stratégies tout en respectant les valeurs de paramètres qui nous semblent optimales, nous avons procédé dans ces expériences à 100 croisements, 8 reproductions par croisement et un crossing-over par reproduction, ce qui nous permet d'explorer 1600 stratégies différentes ; afin d'augmenter la précision de nos résultats, nous avons également réalisé 50 tests lors de chaque expérience. Le but principal de ces expériences est d'étudier l'évolution générale de la proportion des trahisons et des coopérations dans les « **meilleures stratégies des tests** » obtenues dans divers environnements au fur et à mesure que le nombre de parts par affrontement augmente.

Nous avons donc réalisé des affrontements allant de 1 à 15 parts dans chaque environnement avec les valeurs de paramètres citées ci-dessus et en utilisant comme « **stratégies à tester de départ** » que les stratégies GENTIL et MECHANT ; les

proportions de ces gènes dans « **les meilleures stratégies des tests** » se trouvent à l'annexe.

4.2.2.1. Remarques générales concernant les résultats d'expérience

On peut observer dans les affrontements d'une part, donc dans la situation du dilemme du prisonnier simple, que la proportion du gène MECHANT est de 1 et donc celle de GENTIL de 0 en moyenne pour les 50 tests ; et ce quelque soit l'environnement stratégique affronté. Ceci est tout a fait normal, puisque pour chaque test, c'est toujours celle ayant obtenu le plus de points qui est la meilleure, et que si, il n'y a qu'une part, il faut trahir pour obtenir le plus de points (section 1.2.1, p 7). Nous avons également pu observer que les gènes des meilleures « **stratégies hybrides** » étaient le plus souvent GENTIL au début et MECHANT à la fin bien que parfois on retrouve un gène GENTIL au milieu de gènes MECHANTS et vice versa.

Dans ce qui suit, nous allons commenter pour chaque environnement stratégique, l'évolution des proportions de GENTIL et MECHANT dans les « **meilleures stratégies des tests** » en relation avec le nombre de coups par affrontement.

4.2.2.2. Environnement stratégique composé de toutes les stratégies du programme

On peut remarquer que très vite, à partir de 2 coups par affrontement, la proportion du gène GENTIL atteint 0.5 (Figure 16), et que cette proportion semble se stabiliser autour de 0.53 comme nous le confirme un affrontement de 200 coups (**Annexe 9**, p 104)

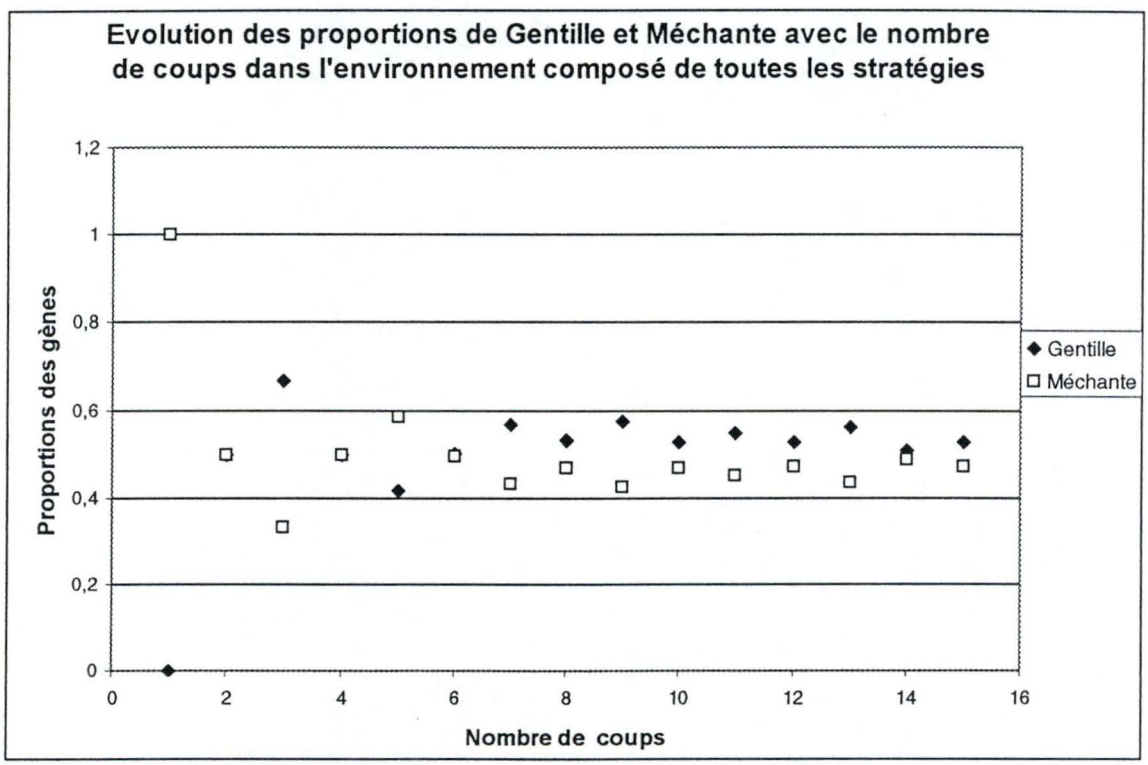

Figure 16 : évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui composé de toutes les stratégies du programme

4.2.2.3. Environnement stratégique composé des stratégies pouvant prendre l'initiative de trahir

Tant que les affrontements ne dépassent pas 3 parts, on ne retrouve que des gènes MECHANTS ; par après la proportion des gènes MECHANTS semble se stabiliser entre 0.6 et 0.7. On peut signaler que MEFIANT, MAJ_DUR, ainsi que LUNATIQUE font partie de cet environnement, les gènes GENTILS permettent donc d'encourager MAJ_DUR et MEFIANT à jouer la coopération et donc d'augmenter le score des stratégies d'affrontement qui les contiennent. Pour l'affrontement de 200 coups, la proportion des MECHANT est de 0.6 (**Annexe 9**, p 104)

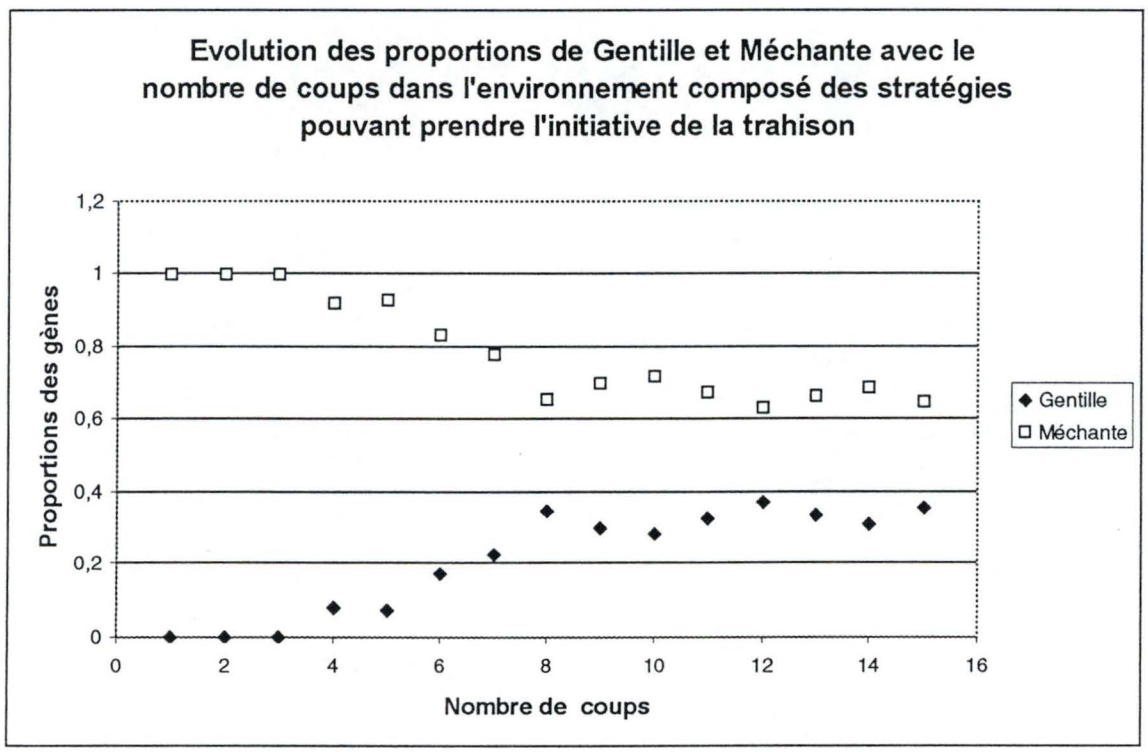

Figure 17 : évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui composé des stratégies pouvant prendre l'initiative de trahir

4.2.2.4. Environnement stratégique composé de stratégies ne prenant jamais l'initiative de la trahison

Dès que les affrontements dépassent deux coups, la proportion des GENTILS atteint 0.5, et semble ensuite se stabiliser autour de 0.65, 0.7, ce que confirme une expérience faite avec un affrontement de 200 coups (**Annexe 9**, p 104).

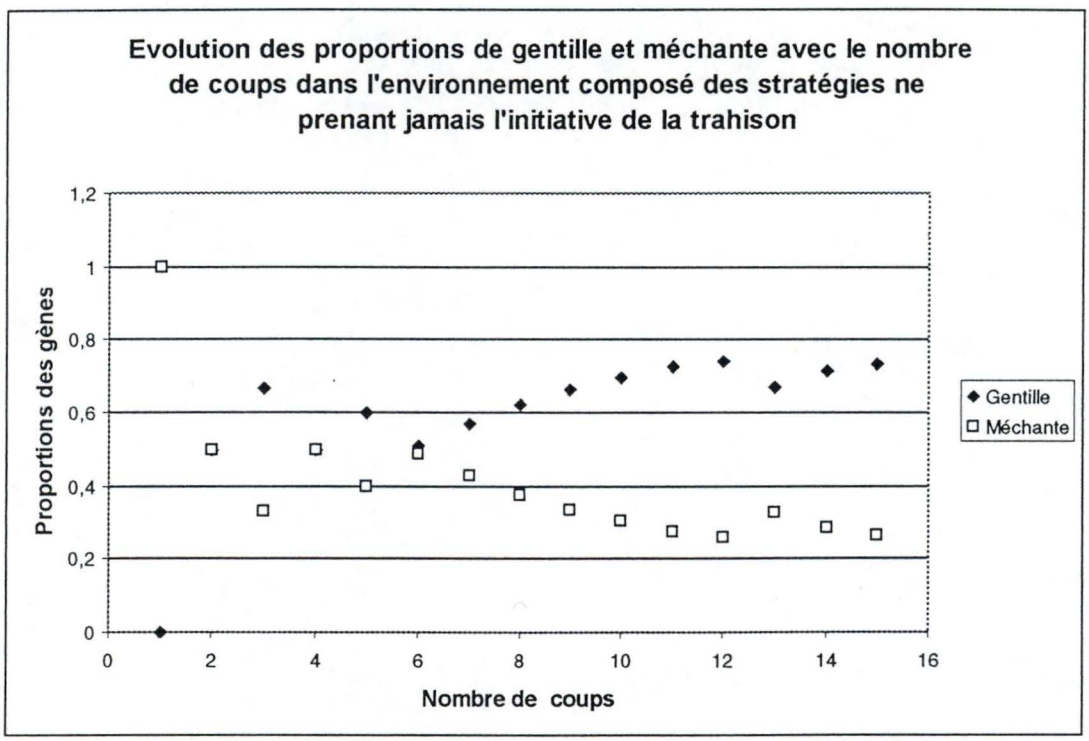

Figure 18: évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui composé de stratégies ne prenant jamais l'initiative de la trahison

4.2.2.5. Environnement stratégique composé de stratégies à jeux indépendants de celui de l'adversaire

Comme on pouvait s'y attendre dans un tel environnement, il ne sert à rien de coopérer puisque les stratégies environnementales n'augmenteront pas le nombre de leurs coopérations ultérieures en fonction des coopérations passées. Quel que soit le nombre de parts par affrontement, on a toujours une proportion de MECHANT qui est de 1.

Cependant, il se pourrait qu'une « **stratégie hybride** » contenant un gène GENTIL et le reste des gènes MECHANTS fasse un meilleur score qu'une stratégie composée uniquement de gènes MECHANTS, bien qu'un gène MECHANT fera forcément de plus beaux scores : au pire 5 points de plus que le gène GENTIL dans la confrontation généralisée avec l'environnement composé des stratégies à jeux indépendants de celui de l'adversaire (Tableau 10, p 60). En effet, les gènes MECHANTS de la stratégie contenant le gène GENTIL pourront combler cette différence par rapport à la stratégie ne contenant que des gènes MECHANTS si ils ont beaucoup plus de chance face à la stratégie LUNATIQUE.

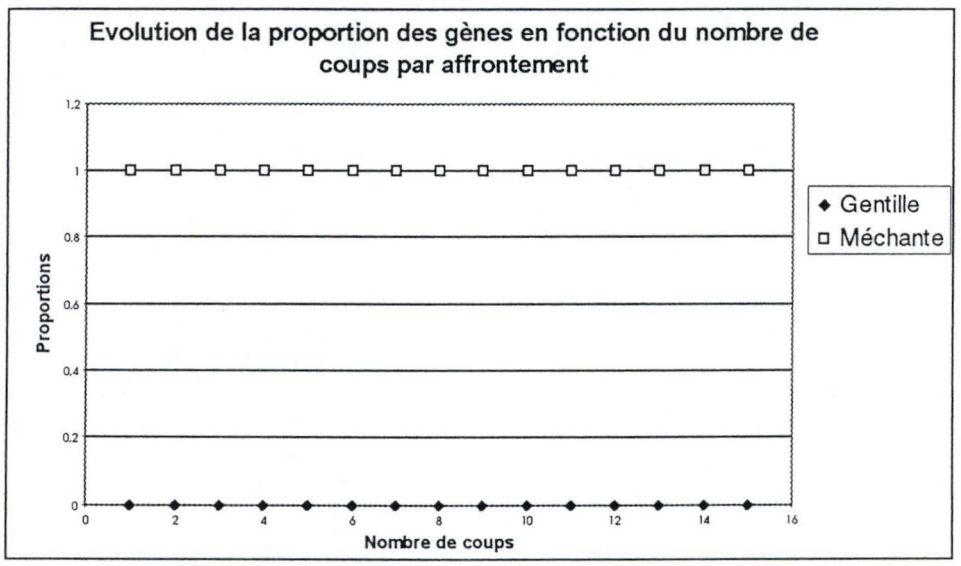

Figure 19 : évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui de stratégies à jeux indépendant de celui de l'adversaire

4.2.2.6. Environnement stratégique composé de stratégies à jeux dépendants de celui de l'adversaire

Comme précédemment, on peut voir qu'à partir d'affrontements de deux parts, la proportion de GENTIL atteint 0.5 et ensuite semble se stabiliser à 0.8 (Figure 20) comme on peut le voir pour un affrontement durant 200 parts (**Annexe 9**, p 104).

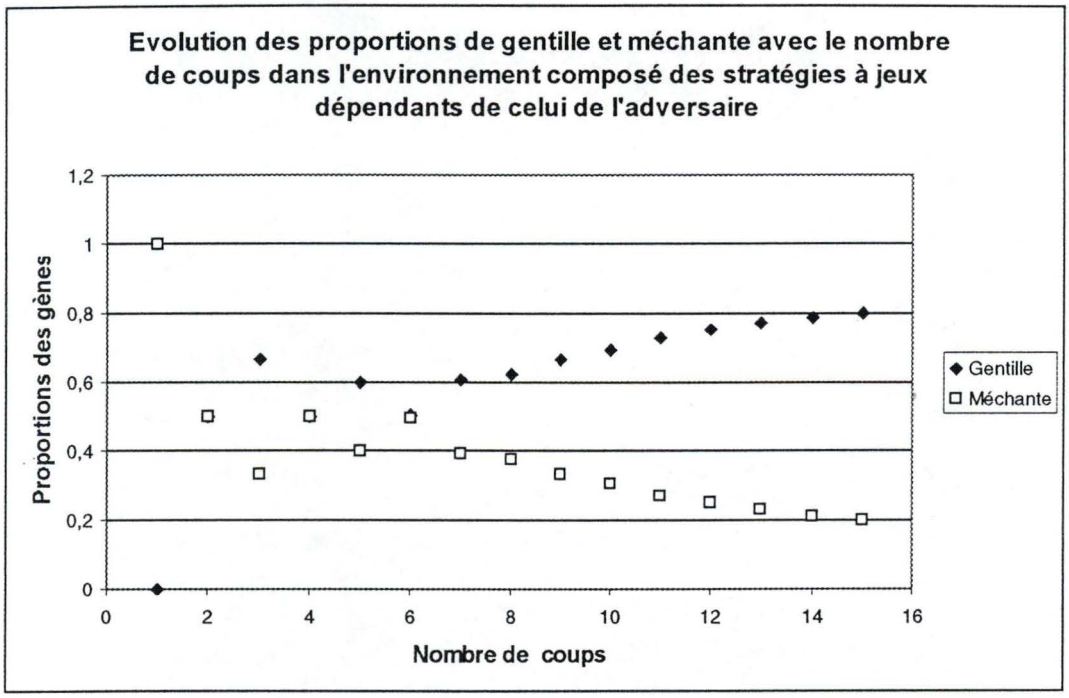

Figure 20 : évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui de stratégies à jeux dépendants de celui de l'adversaire.

4.2.2.7. Environnement stratégique composé de stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux

Tant que les affrontements ne dépassent pas 3 parts, on ne retrouve que des gènes MECHANTS ; ensuite, la proportion de GENTILS se stabilise à 0.52, 0.54 à partir du moment où le nombre de parts par affrontement est de 6.

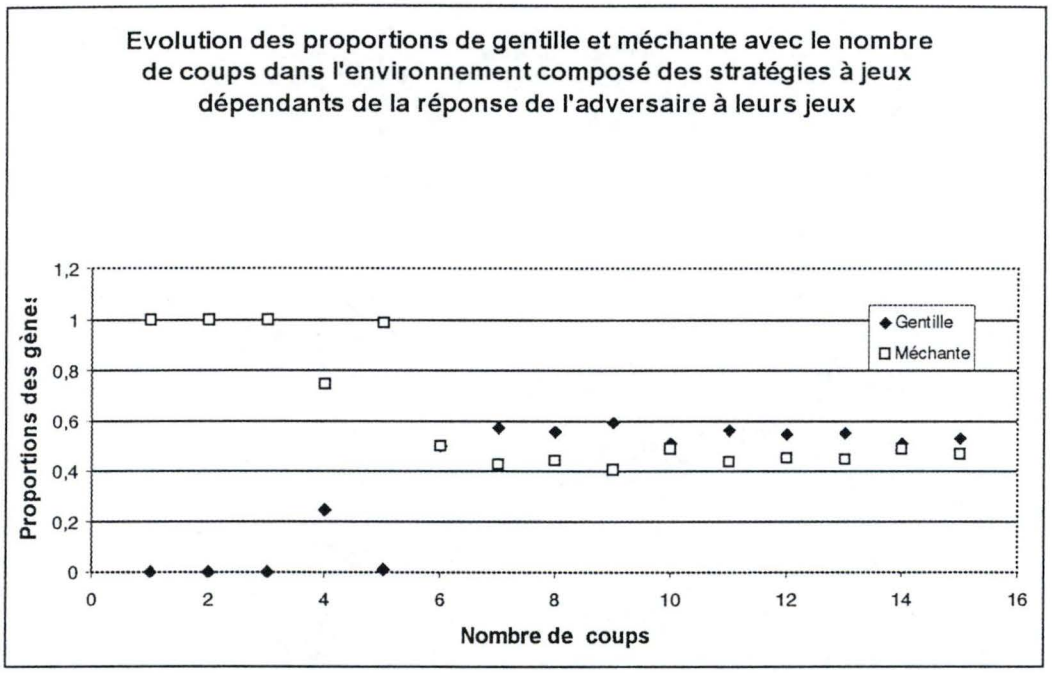

Figure 21 : évolution de la proportion de GENTIL et MECHANT dans les meilleures stratégies au fur et à mesure que le nombre de coups augmente par affrontement ; les paramètres constants sont : 50 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction, et on fait varier le nombre de coups par affrontement. L'environnement stratégique affronté est celui de stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux.

4.2.2.8. commentaires sur les résultats

Les proportions de GENTIL et de MECHANT dans les meilleures stratégies semblent se stabiliser définitivement pour des affrontements durant plus de 8 à 10 coups en général, et ceci quel que soit l'environnement affronté ; lorsque l'environnement est composé des stratégies à jeux dépendants de celui de l'adversaire, il faut attendre des affrontements de 15 coups.

Une autre remarque importante, est qu'il n'y a pas d'environnement contre lequel la proportion des GENTILS est de 1 quelle que soit la durée de l'affrontement. On pourrait penser en voyant la Figure 20 (p 67) que la proportion des GENTILS augmenterait encore dans l'environnement des stratégies à jeux dépendants de celui de l'adversaire. Cependant, nous avons réalisé un affrontement de 200 coups pour lequel nous avons pu constater que cette proportion se situait toujours autour de 0.8 (**Annexe 9**, p 104). Il est évident que la trahison permet aux stratégies d'améliorer leur score, le tout est de bien doser cette trahison et de choisir « judicieusement » l'endroit de cette trahison. Par exemple, trahir lors du premier coup d'un long affrontement est très mauvais face à une « **stratégie simple** » à comportement RANCUNIER, et à partir du moment où l'on trahit une telle stratégie, il ne sert plus à rien d'essayer de renouer avec elle une « amitié », puisqu'elle ne coopérera plus.

De plus, on peut voir à l'**Annexe 1**, p 92 où est représentée la stratégie de 200 gènes obtenue dans l'environnement composé des stratégies à jeux dépendants de celui de l'adversaire, qu'en général les gènes GENTILS se trouvent en début de chromosome, tandis que les gènes MECHANTS se trouvent en fin de chromosome. Cependant, entre le gène 144 et le gène 297, la proportion des gènes majoritaires dans la « **stratégie moyenne** » n'étant plus de 1, on peut penser que les gènes GENTILS et MECHANTS se trouvent mélangés entre ces gènes.

Afin de voir si dans d'autres environnements, les gènes GENTILS et MECHANTS pouvaient à certains endroits se mélanger, nous avons recommencé pour ces environnements des expériences de 20 tests, 200 coups, 100 croisements, 8 reproductions par croisement et 1 crossing-over par reproduction. Les résultats de ces expériences se trouvent à l'**Annexe 2**, (p 92) où nous avons repris la meilleure

stratégie de chaque expérience. On peut remarquer que pour les divers environnements, les gènes GENTILS et MECHANTS ne se retrouvent pas mélangés, à partir de l'endroit du premier gène MECHANT, jusqu'à la fin on ne retrouve plus de gène GENTIL, excepté dans l'environnement composé de stratégies à jeux indépendants de celui de l'adversaire où l'on retrouve un gène GENTIL au milieu des gènes MECHANTS, nous pensons que cela est dû au phénomène expliqué à la section 4.2.2.5, p 66, §2.

Il se peut cependant, que pour des affrontements bien plus importants, les gènes MECHANTS et GENTILS se retrouvent mélangés dans les meilleures stratégies obtenues, mais comme d'une part, nous connaissons mal les valeurs de paramètres à fixer pour de tels affrontements, et que d'autre part, des affrontements de plus de 200 parts demandent un certain temps d'exécution, nous n'avons pas réalisé de telles expériences.

Une explication possible à ce faible mélange pourrait être le faible taux de crossing-over par affrontement ; cependant, nous avions pris la précaution de vérifier que pour des affrontements durant 200 coups, il valait mieux ne réaliser qu'un crossing-over (section 4.1.1.3, p 53). Dès lors, on peut penser qu'il vaut mieux trahir à partir d'un moment donné qui dépend de l'environnement affronté et du nombre de parts de l'affrontement ; cependant, en pratique ce nombre n'est pas connu. Dans notre algorithme, les stratégies à tester de départ ne le connaissent pas non plus, mais nous sélectionnons les meilleures stratégies d'affrontement apparues au hasard ; or à certaine(s) part(s) cela peut rapporter plus d'être méchant, notamment à la dernière part, les stratégies qui trahissent au(x) bon(s) endroit(s) obtiendront donc un plus beau score.

4.2.3. Conclusions

Dans les environnements que nous avons étudiés, il existerait donc un endroit à partir duquel il conviendrait de trahir jusqu'à la fin de l'affrontement et avant cette trahison, il faudrait coopérer. On pourrait expliquer ceci par la présence dans divers environnements : de stratégies qui trahissent au-dessus d'une certaine proportion de trahison (SEUIL, SEUIL_REN, MAJ_MOU, MAJ_DUR, ...) et de stratégies qui ne pardonnent pas (RANCUNIER).

4.3. Robustesse des stratégies créées

Afin de tester la « **robustesse** » (section 1.4.2.4, p 18) des meilleures stratégies et des stratégies d'affrontement moyennes obtenues au moyen des algorithmes génétiques dans chaque type d'environnement, nous avons réalisé des confrontations écologiques de ces stratégies dans tous les types d'environnement. Dans un premier temps, nous avons utilisé comme valeurs de paramètre : 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction et nous avons réalisé des affrontements de 10, 20, 30, 40, 50 et 100 coups. Ensuite, nous avons essayé d'obtenir des stratégies « **robustes** » à partir de l'environnement qui semblait le plus apte à produire de telles stratégies, pour toutes les durées d'affrontement précédemment citées. Pour ce faire, nous avons augmenté le nombre de croisements puisque selon nos observations, il permet d'accroître le score moyen des meilleures stratégies obtenues ; bien que les stratégies obtenant un meilleur score lors d'une confrontation généralisée ne soient pas nécessairement plus robustes que d'autres.

Nous avons décidé de considérer trois cas : si les stratégies meurent (M), gagnent (G) ou survivent (S) à une confrontation écologique. A la fin d'une confrontation écologique, une stratégie est considérée comme morte si son effectif est 0, elle est considérée comme gagnante si il n'y a pas d'autre stratégie ayant un effectif plus grand que le sien, et on dit qu'une stratégie survit, si son effectif est stabilisé, mais qu'il est en dessous de l'effectif le plus important.

Dans ce qui suit, nous allons commenter les résultats de la meilleure stratégie et de la stratégie moyenne obtenue par les algorithmes génétiques pour les 6 environnements.

4.3.1. Environnement composé de toutes les stratégies

Nous observons en général que la « **stratégie moyenne** » et la « **meilleure stratégie de l'expérience** » obtenues au moyen des algorithmes génétiques meurent à peu près autant de fois qu'elles gagnent lors d'une confrontation écologique pour les 6 environnements différents (M = 33, G = 35, S = 4). Le détail des tests se trouve au Tableau 11 ; on peut voir que les stratégies issues des algorithmes génétiques meurent surtout dans l'environnement « qui les a fait naître », dans l'environnement composé des stratégies ne prenant jamais l'initiative de trahir et dans l'environnement des stratégies à jeux dépendants de celui de l'adversaire.

affrontement	stratégie	env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	G	M	G	M	G	G
	la moyenne	G	M	G	M	G	G
20 coups	la meilleure	M	G	M	G	M	G
	la moyenne	M	G	M	G	M	G
30 coups	la meilleure	M	G	M	G	M	G
	la moyenne	M	G	M	G	M	G
40 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	G	G	G	G	G
50 coups	la meilleure	M	G	M	G	M	M
	la moyenne	M	G	M	G	M	M
100 coups	la meilleure	G	G	S	G	G	S
	la moyenne	G	M	S	G	G	S

Tableau 11 : stratégies issues de l'environnement composé de toutes les stratégies du programme ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.2. Environnement composé des stratégies pouvant prendre l'initiative de trahir

Les stratégies issues de cet environnement gagnent beaucoup plus souvent qu'elles ne meurent (M = 8, G = 54, S = 10) ; il n'y a que lors d'une confrontation sur 8 qu'elles disparaissent. Au Tableau 12, on peut remarquer qu'elles meurent principalement lors de confrontation écologique avec toutes les stratégies du programme ; cependant, même dans cet environnement, elles gagnent autant de fois qu'elles meurent. Aucune stratégie ne parvient cependant à gagner toutes les confrontations écologiques.

affrontement	stratégie	env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	G	M	G	M	G	G
	la moyenne	M	M	G	M	G	G
20 coups	la meilleure	G	G	S	G	G	S
	la moyenne	G	G	S	G	G	S
30 coups	la meilleure	M	G	M	G	M	G
	la moyenne	M	G	G	G	G	G
40 coups	la meilleure	G	G	S	G	G	S
	la moyenne	G	G	S	G	G	S
50 coups	la meilleure	M	G	G	G	G	G
	la moyenne	M	G	M	G	M	M
100 coups	la meilleure	G	G	S	G	G	S
	la moyenne	M	G	M	G	M	G

Tableau 12 : stratégies issues de l'environnement composé des stratégies pouvant prendre l'initiative de la trahison ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.3. Environnement composé des stratégies ne prenant jamais l'initiative de trahir

Les stratégies issues de cet environnement meurent deux fois plus souvent qu'elles gagnent dans les confrontations écologiques (M = 49, G = 23) ; et elles gagnent surtout dans les confrontations écologiques avec des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux (Tableau 13). On peut également noter qu'elles font un très mauvais score dans une confrontation écologique avec l'environnement stratégique qui les a fait naître.

		env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	G	M	G	M	G	G
	la moyenne	M	M	G	M	G	G
20 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
30 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	G	M	G	M	G
40 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
50 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
100 coups	la meilleure	M	G	M	G	M	G
	la moyenne	M	G	M	G	M	G

Tableau 13 : stratégies issues de l'environnement composé des stratégies ne prenant jamais l'initiative de la trahison ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendanst de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.4. Environnement composé des stratégies à jeux indépendants de celui de l'adversaire

Les stratégies issues de cet environnement ne gagnent des confrontations écologiques que dans l'environnement des stratégies à jeux indépendants de celui de l'adversaire et dans celui des stratégies pouvant prendre l'initiative de trahir (Tableau 14), ces deux environnements possédant beaucoup de stratégies en commun (Tableau 10, p 60).

affrontement	stratégie	env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	M	M	M	G	M	M
	la moyenne	M	M	M	G	M	M
20 coups	la meilleure	M	G	M	G	M	M
	la moyenne	M	G	M	G	M	M
30 coups	la meilleure	M	G	M	G	M	M
	la moyenne	M	G	M	M	M	M
40 coups	la meilleure	M	M	M	G	M	M
	la moyenne	M	G	M	M	M	M
50 coups	la meilleure	M	M	M	M	M	M
	la moyenne	M	G	M	G	M	M
100 coups	la meilleure	M	M	M	G	M	M
	la moyenne	M	M	M	G	M	M

Tableau 14 : stratégies issues de l'environnement composé des stratégies à jeux indépendants de celui de l'adversaire ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.5. Environnement composé des stratégies à jeux dépendants de celui de l'adversaire

Les stratégies issues de cet environnement gagnent surtout dans l'environnement des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux, et meurent beaucoup dans les autres environnements (M = 50, G = 22) comme on peut le voir au Tableau 15.

affrontement	stratégie	env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	G	G	G	M	G	G
	la moyenne	M	M	G	M	G	G
20 coups	la meilleure	M	M	M	M	M	G
	la moyenne	G	M	G	M	G	M
30 coups	la meilleure	G	M	G	M	G	G
	la moyenne	M	M	M	M	M	G
40 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	M
50 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
100 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G

Tableau 15 : stratégies issues de l'environnement composé des stratégies à jeux dépendants de celui de l'adversaire ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.6. Environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leur propre jeu

De tous les environnements, ce sont les stratégies issues de cet environnement qui gagnent le moins souvent dans les confrontations écologiques (M = 61, G = 11). Le détail des test se trouve dans le tableau ci-dessous. On peut cependant rappeler que cet environnement n'est composé que de 4 stratégies de notre programme (Tableau 10, p 60).

affrontement	stratégie	env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	M	M	M	M	M	M
	la moyenne	M	M	M	M	M	M
20 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	M
30 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
40 coups	la meilleure	M	G	M	G	M	G
	la moyenne	M	G	M	G	M	G
50 coups	la meilleure	M	M	M	M	M	M
	la moyenne	M	M	M	M	M	M
100 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G

Tableau 16 : stratégies issues de l'environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leur propre jeux; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.7. Commentaires à propos des confrontations écologiques

En ce qui concerne la « **stratégie moyenne** » et la « **meilleure stratégie de l'expérience** », il ne semble pas que l'une d'elles soit en général plus robuste que l'autre lorsque l'on compare leurs résultats (Tableau 17).

Dans nos expériences, c'est donc l'environnement des stratégies pouvant prendre l'initiative de la trahison qui semble le plus propice à créer des stratégies « **robustes** ». On aurait pu s'attendre à ce que ce soit plutôt l'environnement composé de toutes les stratégies qui produisent de telles stratégies, or bien que dans ce cas, il y ait presque la moitié des confrontations écologiques qui soient gagnées, nous pouvons penser que ceci est en partie dû aux stratégies pouvant prendre l'initiative de trahir, qui font également partie de cet environnement.

Dans nos expériences, excepté l'environnement composé de toutes les stratégies du programme, les autres environnements produisent des stratégies qui en moyenne gagnent plus de deux fois moins de confrontations écologiques que celles produites dans les mêmes conditions par l'environnement composé des stratégies pouvant prendre l'initiative de la trahison (Tableau 17).

La raison pour laquelle c'est principalement l'environnement composé des stratégies pouvant prendre l'initiative de la trahison, qui est propice à la création de stratégies robustes ne nous est pas connue avec certitude. Cependant, nous pensons que la présence dans cet environnement de stratégies « **méchantes** » est avantageuse car cela permet d'augmenter la réactivité des nouvelles stratégies. De plus, deux stratégies à jeux dépendants (MAJ_DUR et MEFIANT) de celui de l'adversaire permettent de récompenser les stratégies qui pardonnent le premier coup de l'affrontement. En effet, nous avons refait la même expérience en remplaçant MAJ_DUR par MAJ_MOU et MEFIANT par TIT_FOR_TAT, et dans ce cas nous avons obtenu une stratégie moyenne qui était RANCUNIER avec une proportion de 0,9 pour tous ses gènes moins « robuste » que celles obtenues dans l'environnement des stratégies pouvant prendre l'initiative de trahir.

Stratégies environnementales qui ont fait naître la meilleure stratégie et la stratégie moyenne		M	G	S
toutes	la meilleure	19	15	2
	la moyenne	14	20	2
pouvant prendre l'initiative de trahir	la meilleure	0	30	6
	la moyenne	8	24	4
ne prenant jamais l'initiative de trahir	la meilleure	25	11	0
	la moyenne	24	12	0
à jeux indépendants de celui de l'adversaire	la meilleure	29	7	0
	la moyenne	28	8	0
à jeux dépendants de celui de l'adversaire	la meilleure	23	13	0
	la moyenne	27	9	0
à jeux dépendants de la réponse de l'adversaire à son jeu	la meilleure	30	6	0
	la moyenne	31	5	0

Tableau 17 : pour les 6 environnements stratégiques de la première colonne, nous avons repris le nombre de fois que meurent (M) la meilleure stratégie et la stratégie moyenne issues de chaque environnement, ainsi que le nombre de fois où elles gagnent (G) ou survivent (S) à la confrontation écologique.

Ces résultats sont encourageants, car ils nous permettent de penser que pour éditer des stratégies « **robustes** », il n'y a pas besoin de connaître toutes les stratégies possibles qui sont évidemment en nombre infini. Cependant, même lorsque les stratégies sont issues de l'environnement composé des stratégies pouvant prendre l'initiative de la trahison, aucune des stratégies ne parvient à gagner toutes les confrontations écologiques ; certaines cependant ne meurent dans aucune confrontation écologique (ce qui n'est peut-être déjà pas si mal !).

Afin de voir si dans l'environnement composé des stratégies pouvant prendre l'initiative de la trahison, il était possible d'obtenir des stratégies « **robustes** » qui gagnent dans chaque environnement, nous avons réalisé les mêmes expériences en faisant varier le nombre de croisements.

Comme on peut le voir sur le Tableau 18 (p 80), nous avons réussi à trouver des stratégies qui gagnent dans toutes les confrontations écologiques ; c'est-à-dire qu'après stabilisation de l'effectif des différentes stratégies, on ne retrouve pas de stratégie ayant un effectif plus grand que le sien, et ce quel que soit l'environnement stratégique affronté lors de la confrontation écologique.

Il ne semble pas y avoir de relation nette entre le score fait par une stratégie dans l'environnement qui l'a fait naître et sa robustesse, il semblerait cependant que les stratégies obtenant un très beau score soient très peu « **robustes** » (Tableau 18) ;

en effet, elles gagnent souvent la confrontation écologique contre l'environnement qui les a fait naître, mais rarement lorsqu'il s'agit d'un autre environnement. On pourrait expliquer cette faible robustesse par le fait que ces stratégies sont trop bien adaptées à l'environnement stratégique qui les a fait naître, et sont donc trop dépendantes des stratégies qui composent cet environnement. Lorsque la proportion de ces dernières ou de certaines d'entre elles diminue dans l'environnement stratégique qu'elles affrontent, elles obtiennent beaucoup moins de points et sont plus enclines à disparaître.

	score	env 1	env 2	env 3	env 4	env 5	env 6
stratégie 1	834	G	G	G	G	G	G
stratégie 2	844	G	M	G	M	G	G
stratégie 3	854	M	M	M	M	M	M
stratégie 4	875	G	M	G	G	G	G
stratégie 5	883	G	M	G	G	G	G
stratégie 6	891	G	G	G	G	G	G
stratégie 7	894	M	G	M	G	M	G
stratégie 8	926	M	M	M	M	M	M
stratégie 9	926	M	G	M	G	M	G
stratégie 10	935	M	G	M	M	M	M
stratégie 11	945	M	G	M	G	M	G

Tableau 18 : 11 stratégies issues de l'environnement composé des stratégies pouvant prendre l'initiative de trahir ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leur jeux). Ces 11 stratégies sont en fait les meilleures stratégies de 11 expériences différentes obtenues en faisant varier le nombre de croisements et en faisant 10 tests, 50 coups, 8 reproductions, 1 crossing-over. Le score est le score qu'elles obtiennent lors d'une confrontation généralisée dans l'environnement composé des stratégies pouvant prendre l'initiative de la trahison (env 2). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

Comme nous ne pouvons garantir la « **robustesse** » d'une stratégie sur base de son score, la seule technique dont nous disposons pour essayer d'évaluer cette « **robustesse** » est de réaliser avec cette stratégie des confrontations écologiques dans divers environnements. Cette technique est critiquable, puisqu'il existe une infinité d'environnements stratégiques possibles ; à ce propos, il serait intéressant d'essayer de sélectionner un nombre fini d'environnements et de stratégies avec lesquels on testerait la robustesse et qui refléteraient bien les environnements stratégiques généralement rencontrés, sachant qu'il n'existe probablement pas de stratégie « **robuste** » dans l'absolu.

Lors d'expériences ultérieures, nous avons réussi à faire naître dans l'environnement composé des stratégies pouvant prendre l'initiative de la trahison, des stratégies qui gagnaient la confrontation écologique contre chacun de nos environnements, et ceci pour des affrontements de 10, 20, 30, 40, 50 et 100 coups. Cependant, nous ne sommes pas parvenu à obtenir pour des affrontements de 5 parts des stratégies qui gagnaient les confrontations écologiques dans tous les environnements. Une autre remarque importante, est que ces stratégies étaient « **robustes** » lors de ces expériences ; en effet, il se peut que lors d'une expérience une stratégie perde une confrontation écologique et que lors d'une autre expérience elle survive ou gagne cette même confrontation écologique, ceci étant dû à la présence du « **comportement** » LUNATIQUE.

4.3.8. Analyse des gènes

A l'**Annexe 1**, p 92, on peut voir les stratégies qui ont gagné toutes les confrontations écologiques. On peut remarquer que, quelle que soit la durée de l'affrontement, on a une séquence d'au moins 4 gènes au début du chromosome qui sont dans nos expériences toujours des stratégies ne prenant jamais l'initiative de la trahison. Au milieu du chromosome, on a toujours une séquence de T_F_T_DUR dont la longueur s'allonge avec le nombre de parts de l'affrontement ; au-dessus de 20 parts par affrontement, la proportion de T_F_T_DUR atteint 0.5 (Tableau 19,p 81). En fin d'affrontement, on a le plus souvent une séquence de gènes RANCUNIERS dont la proportion atteint 0.3 en général (Tableau 19,p 81).

	10 coups	20 coups	30 coups	40 coups	50 coups	100 coups	200 coups
PROB_MEME	0,2	0,1	0,1	0,1	0	0,04	0
PROB_CONTINUE	0,2		0,03333333	0	0	0,01	0
T_F_T_DUR	0,2	0,45	0,63333333	0,9	0,52	0,67	0,97
RANCUNIER	0,1	0,35	0,23333333	0	0,36	0,27	0
PAV_CINQ	0	0	0	0	0	0,01	0,025
GENTIL	0	0	0	0	0,12	0	0
TESTCONTINUE	0	0,1	0	0	0	0	0
MECHANT	0,3	0	0	0	0	0	0
PERIOD_MECH	0	0	0	0	0	0	0,005

Tableau 19 : proportion des gènes dans les stratégies d'affrontement qui gagnent toutes les confrontations écologiques, pour les différentes longueurs d'affrontement.

Nous avons ensuite essayé d'obtenir une stratégie robuste pour un affrontement de 200 parts, nous avons eu beaucoup de difficultés à trouver une telle stratégie ; cependant nous y sommes parvenus. Le détail de cette stratégie est présenté dans le tableau ci-dessous.

Comme on peut le voir à l'**Annexe 4**, p 100, la proportion du gène T_F_T_DUR au sein de cette stratégie est très élevée : 0,97. On peut également remarquer que la séquence de départ est toujours présente et qu'elle a toujours plus ou moins la même longueur ici : 5 gènes, tandis que la séquence de la fin du chromosome s'est considérablement rétrécie : 1 gène.

Lors d'une expérience ultérieure, nous avons testé la robustesse des stratégies générées à partir de gènes MECHANTS et GENTILS dans l'environnement composé de toutes les stratégies du programme. Comme on pouvait s'y attendre, de telles stratégies perdent la plupart des confrontations écologiques, excepté face à l'environnement stratégique composé des stratégies à jeux dépendant de la réponse de l'adversaire à leurs jeux.

		env 1	env 2	env 3	env 4	env 5	env 6
10 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
20 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
30 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
40 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
50 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G
100 coups	la meilleure	M	M	M	M	M	G
	la moyenne	M	M	M	M	M	G

Tableau 20 : stratégies à gènes uniquement GENTILS et MECHANTS issues de l'environnement composé de toutes les stratégies du programme ; pour chaque affrontement, nous avons réalisé 10 tests, 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction. Ces stratégies ont réalisé une confrontation écologique dans les 6 environnements stratégiques (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant prendre l'initiative de la trahison, env 3 = environnement composé des stratégies ne prenant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnements composé des stratégies à jeux dépendant de la réponse de l'adversaire à leurs jeux). Un G signifie que la stratégie gagne la confrontation écologique, un M signifie qu'elle meure et un S signifie qu'elle parvient à survivre.

4.3.9. Conclusion

Il semble qu'il existe des « **stratégies robustes** » du moins dans nos environnements, et ces stratégies sont le plus souvent générées à partir de notre environnement stratégique composé des stratégies pouvant prendre l'initiative de la trahison. Ces stratégies semblent constituées de 3 séquences homogènes de gènes : en début de chromosome, on observe une séquence de 5 ou 6 gènes dont nous pensons qu'ils jouent en grande majorité la coopération, ensuite, nous avons une séquence de gènes T_F_T_DUR et en fin de chromosome on observe en général une séquence de gènes dont nous pensons qu'ils jouent majoritairement la trahison (**Annexe 3**, p 99), ces deux dernières séquences semblent s'allonger avec le nombre de parts. Néanmoins, nous pensons que la dernière séquence de gènes n'est pas indispensable pour que la stratégie soit robuste, puisque certaines stratégies ne possédant pas cette séquence sont robustes (**Annexe 4**, p 100).

4.4. Comparaison des résultats obtenus avec ceux de Delahaye et Mathieu (1992)

Delahaye et Mathieu (1992) réalisèrent des confrontations écologiques avec les douze stratégies présentées dans le Tableau 3 (p 13). Les résultats qu'ils avaient obtenus leur avaient permis alors de confirmer la proposition de Axelrod selon laquelle TIT_FOR_TAT est une bonne stratégie. Nous avons réalisé quatre expériences dans des conditions proches, mais en générant de nouvelles stratégies au moyen de l'algorithme génétique. Pour ces expériences, nous avons choisi d'utiliser toutes les « **stratégies à tester** » que nous avons confrontées à l'environnement des stratégies utilisées par Delahaye et Mathieu, à l'exception de SONDEUR, qui rappelons-le n'a pas été implémenté (section 3.3, p 40). Les expériences ont été réalisées avec les paramètres suivants : 100 coups, 30 croisements, 6 reproductions, 1 crossing-over et 10 tests.

Ces expériences nous ont permis de découvrir des stratégies meilleures que TIT_FOR_TAT dans cet environnement. En effet, toutes les stratégies formées se classent en première position en fin de confrontation écologique (**Annexe 5**, p 101).

Ces stratégies ont toutes des caractéristiques proches, ce qui permet de tirer les propriétés d'une stratégie gagnant une confrontation écologique dans l'environnement de Delahaye et Mathieu. Ces caractéristiques sont les suivantes :

- une stratégie réactive
- une stratégie qui réagit de plus en plus sévèrement
- une stratégie qui ne prend l'initiative de trahir que dans les tous derniers coups

On peut également remarquer que lors des tests de confrontation écologique de ces nouvelles stratégies dans les autres environnements types, celles-ci ont presque toujours été classées premières à la fin de la confrontation, et toutes ont survécu (**Annexe 6**, p 103). Ceci tend à prouver qu'il existerait des stratégies qui soient robustes dans tous les environnements.

On remarque également que ces stratégies se rapprochent de la stratégie LA_MEILEURE trouvée par Delahaye et Mathieu (section 1.4.2.5, p 19), à l'exception qu'elles ne renoncent jamais, et qu'elles prennent parfois l'initiative de trahir dans les derniers gènes.

4.5. Discussion sur les expériences

Il convient de relativiser le terme de « **robustesse** » employé ici, car les confrontations écologiques sont réalisées pour un affrontement dont la durée est la même que la durée des affrontements qui ont fait naître la « **meilleure stratégie de l'expérience** » et la « **stratégie moyenne** ». Or, on remarque que les derniers gènes de certaines stratégies obtenues peuvent être MECHANT (**Annexe 3**, p 99), dans une confrontation écologique durant par exemple le double de temps, on ne peut donc pas penser rajouter à la fin de la séquence de gènes de la stratégie, cette même séquence. Cela ne produirait probablement pas une stratégie « **robuste** » car elle trahirait certainement trop tôt.

De plus, certaines stratégies peuvent faire le maximum de points lors de la première part d'une confrontation écologique et quand même disparaître au cours de cette dernière, et n'avoir aucune chance de survie dans d'autres confrontations écologiques. A ce propos, il serait intéressant de voir si la stratégie « LA_MEILLEURE » (section 1.4.2.3, p 17) de J. P. Delahaye & Ph. Mathieu (1993)

gagnerait une confrontation écologique dans d'autres environnements que celui composé des stratégies de leur concours.

D'un autre coté, une stratégie obtenant un faible score dans une confrontation écologique avec l'environnement qui l'a fait naître, a souvent tendance à disparaître lors de la confrontation écologique. Il semble que les stratégies qui gagnent toutes les confrontations écologiques soient souvent des stratégies obtenant un score relativement bon, mais pas spécialement excellent ; du moins, pour les stratégies issues de l'environnement composé des stratégies pouvant prendre l'initiative de la trahison.

Nous pensons que le terme de « **robustesse** » devrait principalement s'appliquer aux stratégies pouvant gagner des confrontations écologiques dans un grand nombre d'environnements différents. Rappelons que nous considérons qu'une stratégie gagne une confrontation écologique dans le cas où après stabilisation des effectifs, il n'y a pas d'autre(s) stratégie(s) ayant un effectif plus important que le sien.

4.6. Propositions d'expériences ultérieures

Il serait intéressant de réaliser un module permettant par exemple d'accroître la longueur d'une stratégie obtenue par les algorithmes génétiques de plusieurs manières. On pourrait par exemple supprimer la séquence de gènes terminale (section 4.2.3, p 70); dans le cas où on désire un affrontement deux fois plus long, on doublerait alors cette séquence terminale et on la rajouterait à la fin de la nouvelle stratégie, on boucherait alors le trou avec la séquence centrale (section 4.2.3, p 70) de la stratégie initiale (Figure 22, p 86).

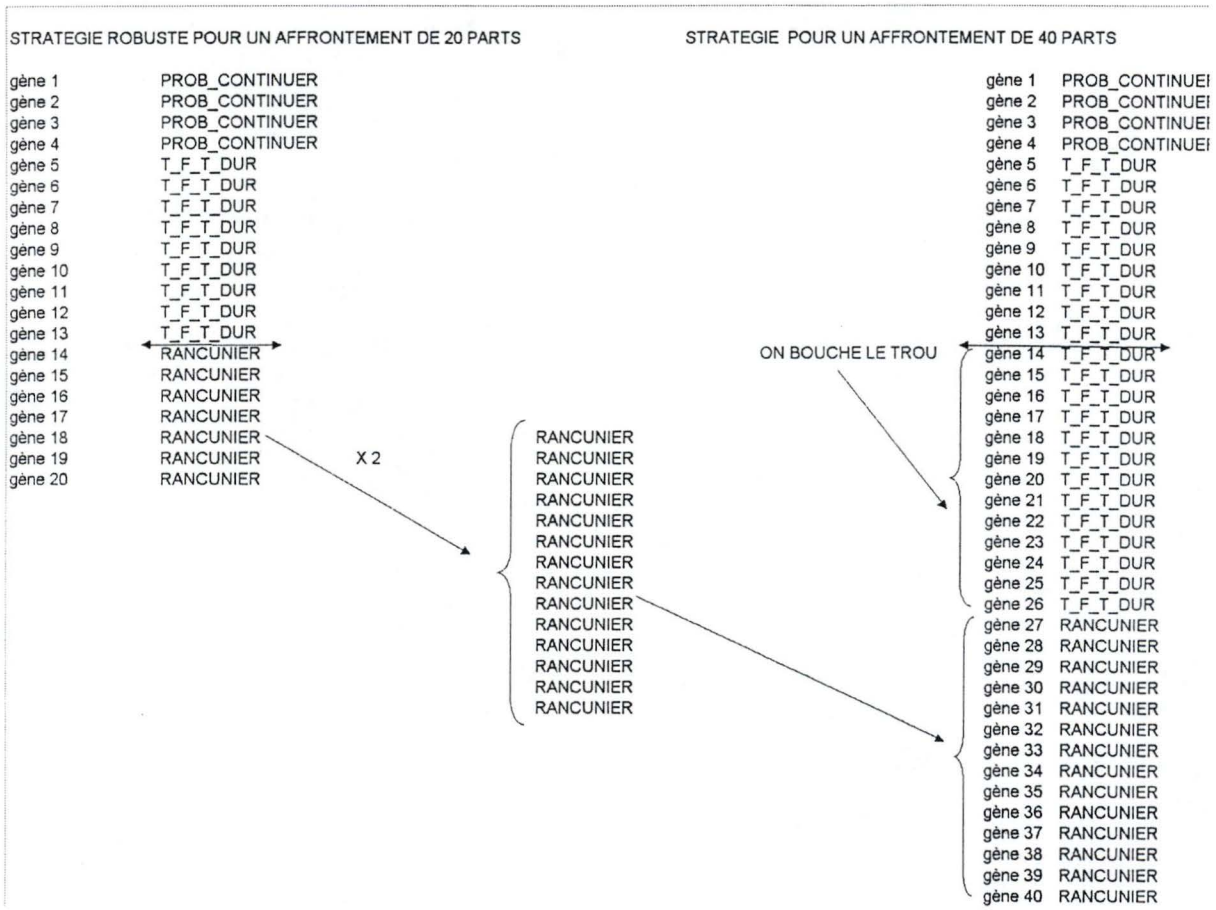

Figure 22 : Processus d'allongement possible d'une stratégie robuste en vue de produire une autre stratégie robuste pour un nombre de parts plus élevé.

On pourrait également réaliser une expérience dans laquelle on se contenterait d'allonger la séquence centrale de gènes.

Si une telle expérience conduisait à la création de stratégies « **robustes** », lorsque le nombre de parts d'un dilemme du prisonnier itéré n'est pas connu à l'avance par chacune des parties, on pourrait envisager de simplement rajouter la séquence centrale et laisser tomber la séquence finale. Dans nos expériences, nous avons pu remarquer que certaines stratégies « **robustes** » n'étaient pas composées de la troisième séquence (section 4.2.3, p 70).

On pourrait également faire des expériences avec seulement certains gènes, rajouter des stratégies ; étudier plus en profondeur les séquences de gènes…

Essayer de trouver un environnement propice à la naissance de stratégies « **robustes** », l'environnement qui semblait le plus adéquat parmi nos environnements était celui des stratégies pouvant prendre l'initiative de la trahison, on pourrait essayer d'améliorer cet environnement.

Pour tester la « **robustesse** » de manière plus précise, il faudrait créer de nouveaux environnements stratégiques.

On pourrait également améliorer notre interface pour permettre de nouvelles possibilités comme : sélectionner n'importe quelle stratégie à tester pour réaliser les confrontations écologiques. Ceci permettrait de tester la robustesse des stratégies de départ et d'autres stratégies n'ayant pas nécessairement obtenu le meilleur score.

Dans l'avenir, il serait également intéressant de voir si il y a toujours moyen de trouver des stratégies « **robustes** » si on change les bénéfices accordés aux joueurs.

CONCLUSIONS GÉNÉRALES

Comme on l'espérait, les algorithmes génétiques nous ont permis de générer des nouvelles stratégies intéressantes. Ainsi, nous avons obtenu des stratégies capables de faire des scores bien au-dessus de ceux des « **stratégies simples** » lors de confrontations généralisées. De plus, certaines stratégies générées résistent très bien aux confrontations écologiques que nous leur avons fait subir dans divers environnements.

Ces stratégies sont en général constituées de 3 séquences homogènes de gènes : il semble que la séquence de début soit assez gentille, la séquence du milieu est souvent fort réactive, tandis que la dernière séquence est plutôt méchante. Néanmoins nous pensons que la dernière séquence de gènes n'est pas indispensable pour que la stratégie soit robuste, puisque certaines stratégies ne possédant pas cette séquence sont robustes (**Annexe 4**, p 100).

Nous pouvons également comparer les caractéristiques de nos stratégies « **robustes** » avec celles que Delahaye et Mathieu avaient soulignés (section 1.4.2.5, p 19).

- *Elle coopère à la première « **part** ».* C'est bien ce que l'on observe dans nos stratégies « **robustes** ».
- *Elle ne prend jamais l'initiative de la trahison c'est une « **gentille** ».* Ceci est vérifié dans nos stratégies « **robustes** », excepté en fin de chromosomes, où on peut observer des gènes MECHANTS.
- *Elle renonce si après 20 parts le score obtenu est inférieur à 1,5.* Nous n'avons pas observé de stratégie « **robuste** » composée du gène SEUIL_REN (le seul utilisant le renoncement). Ceci pourrait s'expliquer par le fait que le renoncement n'était pas définitif, ou que cette stratégie n'obtenait pas assez de points au départ ; son seuil de renoncement (score moyen < 2) n'étant peut-être pas le plus approprié.
- *Elle entre dans une période de punition lorsqu'elle est trahie.*
- *Elle est de plus en plus sévère.* Nous observons que les stratégies « **robustes** » sont composées de gènes permettant une punition dans le cas où l'adversaire trahit. Cependant, cette punition dans la séquence centrale

de gènes ne dépasse pas deux coups, mais dans la séquence finale, la punition est en général plus sévère, notons toutefois, qu'il ne s'agit peut-être que d'une tentative d'exploiter l'adversaire.

REFERENCES BIBLIOGRAPHIQUES

R. AXELROD, 1984. *The Evolution of Cooperation.* Basic Books, Inc. Publishers, New York. Traduction française : *Donnant donnant* Ed. O. Jacob, Paris 1992.

R. AXELROD 1987. *The evolution of strategies in the iterated prisoner's dilemma.* In L. Davis, editor, Genetic Algorithms and Simulated Annealing, chapter 3, pages 32-41. Pitman.

J.P.DELAHAYE & P.MATHIEU, 10/06/1992. *Expériences sur le dilemme itéré des prisonniers*. Université des Sciences et Technologies de Lille. Publication interne n°**233**.

J. P. DELAHAYE & PH. MATHIEU, 05/05/1993. *L'altruisme perfectionné.* Université des Sciences et Technologies de Lille. Publication interne n°**249**.

J. P. DELAHAYE & PH. MATHIEU , 1996. *Our meeting with gradual : a good strategy for the iterated prisoner's dilemma.*

N. FALLETTA, 1985. *Le livre des paradoxes*. Belfond, Paris. Traduction française de « *The Paradoxicon* », Doubleday and Co. New York 1983.

D. GLOVER, 1988. *Solving a Complex Keyboard Configuration Problem Through Generalized Adaptive Search.* In L. Davis, editor, Genetic Algorithms and Simulated Annealing, chapter 2, pages 12-31. Pitman.

D. E. GOLDBERG, 1983. *Computer-aided gas pipeline operation using genetic algorithms and rule learning.* PhD thesis, University of Michigan.

J. H. HOLLAND, 1975. *Adaptation in Natural and Artificial Systems*. MIT Press.

I. C. LERMAN & R. F. NGOUENET, 1995. Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités. Publication interne n°**901**.

POUNDSTONE, 1990. *Les labyrinthes de la raison* : Paradoxes, énigmes et fragilité de la connaissance. Belfond, Paris. Traduction française de « *Labyrinths of the reason* » Anchor Doubleday Publishing Company, New York, 1988.

J. Y. SUH & D. V. GUCHT, july 1987. *Distributed genetic algorithms*. Technical Report 225, Computer Science Departement, Indiana University.

Annexes

Annexes

Annexe 1 : proportions des gènes MECHANTS et GENTILS au sein d'une « **stratégie moyenne** » obtenue en affrontant l'environnement des stratégies à jeux dépendants de celui de l'adversaire et en réalisant 20 tests, 200 coups, 100 croisements, 8 reproductions par croisement et 1 crossing-over par reproduction

	50 premiers	prop	100 premiers	prop	150 premiers	prop	200 premiers	prop
gène 1	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,9
gène 2	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,9
gène 3	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,9
gène 4	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,9
gène 5	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,9
gène 6	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 7	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 8	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 9	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 10	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 11	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,85
gène 12	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 13	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 14	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 15	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 16	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 17	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,75
gène 18	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 19	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 20	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 21	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 22	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 23	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,65
gène 24	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 25	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 26	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 27	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 28	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 29	GENTIL	1	GENTIL	1	GENTIL	1	GENTIL	0,5
gène 30	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 31	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 32	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 33	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 34	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 35	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,55
gène 36	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 37	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 38	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 39	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 40	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 41	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,65
gène 42	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,85

gène 43	GENTIL	1	GENTIL	1	GENTIL	1	MECHANT	0,85
gène 44	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	0,85
gène 45	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	0,85
gène 46	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	0,85
gène 47	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	0,85
gène 48	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	1
gène 49	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	1
gène 50	GENTIL	1	GENTIL	1	GENTIL	0,9	MECHANT	1

Annexe 2 : « meilleures stratégies d'expérience » obtenues dans les différENts environnements pour des affrontements de 200 coups, 20 tests, 100 croisements, 8 reproduction par croisement, 1 crossing-over par reproduction.

environnement : toutes les stratégies du programme

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	GENTIL	GENTIL	MECHANT	MECHANT
gène 2	GENTIL	GENTIL	MECHANT	MECHANT
gène 3	GENTIL	GENTIL	MECHANT	MECHANT
gène 4	GENTIL	GENTIL	MECHANT	MECHANT
gène 5	GENTIL	GENTIL	MECHANT	MECHANT
gène 6	GENTIL	GENTIL	MECHANT	MECHANT
gène 7	GENTIL	GENTIL	MECHANT	MECHANT
gène 8	GENTIL	GENTIL	MECHANT	MECHANT
gène 9	GENTIL	GENTIL	MECHANT	MECHANT
gène 10	GENTIL	GENTIL	MECHANT	MECHANT
gène 11	GENTIL	GENTIL	MECHANT	MECHANT
gène 12	GENTIL	GENTIL	MECHANT	MECHANT
gène 13	GENTIL	GENTIL	MECHANT	MECHANT
gène 14	GENTIL	GENTIL	MECHANT	MECHANT
gène 15	GENTIL	GENTIL	MECHANT	MECHANT
gène 16	GENTIL	GENTIL	MECHANT	MECHANT
gène 17	GENTIL	GENTIL	MECHANT	MECHANT
gène 18	GENTIL	GENTIL	MECHANT	MECHANT
gène 19	GENTIL	GENTIL	MECHANT	MECHANT
gène 20	GENTIL	GENTIL	MECHANT	MECHANT
gène 21	GENTIL	GENTIL	MECHANT	MECHANT
gène 22	GENTIL	GENTIL	MECHANT	MECHANT
gène 23	GENTIL	GENTIL	MECHANT	MECHANT
gène 24	GENTIL	GENTIL	MECHANT	MECHANT
gène 25	GENTIL	GENTIL	MECHANT	MECHANT
gène 26	GENTIL	GENTIL	MECHANT	MECHANT
gène 27	GENTIL	GENTIL	MECHANT	MECHANT
gène 28	GENTIL	GENTIL	MECHANT	MECHANT
gène 29	GENTIL	GENTIL	MECHANT	MECHANT
gène 30	GENTIL	GENTIL	MECHANT	MECHANT
gène 31	GENTIL	GENTIL	MECHANT	MECHANT
gène 32	GENTIL	GENTIL	MECHANT	MECHANT
gène 33	GENTIL	GENTIL	MECHANT	MECHANT
gène 34	GENTIL	GENTIL	MECHANT	MECHANT
gène 35	GENTIL	GENTIL	MECHANT	MECHANT

gène 36	GENTIL	GENTIL	MECHANT	MECHANT
gène 37	GENTIL	GENTIL	MECHANT	MECHANT
gène 38	GENTIL	GENTIL	MECHANT	MECHANT
gène 39	GENTIL	GENTIL	MECHANT	MECHANT
gène 40	GENTIL	GENTIL	MECHANT	MECHANT
gène 41	GENTIL	GENTIL	MECHANT	MECHANT
gène 42	GENTIL	GENTIL	MECHANT	MECHANT
gène 43	GENTIL	GENTIL	MECHANT	MECHANT
gène 44	GENTIL	GENTIL	MECHANT	MECHANT
gène 45	GENTIL	GENTIL	MECHANT	MECHANT
gène 46	GENTIL	GENTIL	MECHANT	MECHANT
gène 47	GENTIL	GENTIL	MECHANT	MECHANT
gène 48	GENTIL	GENTIL	MECHANT	MECHANT
gène 49	GENTIL	MECHANT	MECHANT	MECHANT
gène 50	GENTIL	MECHANT	MECHANT	MECHANT

environnement : stratégies pouvant prendre l'initiative de trahir

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	GENTIL	GENTIL	MECHANT	MECHANT
gène 2	GENTIL	GENTIL	MECHANT	MECHANT
gène 3	GENTIL	GENTIL	MECHANT	MECHANT
gène 4	GENTIL	GENTIL	MECHANT	MECHANT
gène 5	GENTIL	GENTIL	MECHANT	MECHANT
gène 6	GENTIL	GENTIL	MECHANT	MECHANT
gène 7	GENTIL	GENTIL	MECHANT	MECHANT
gène 8	GENTIL	GENTIL	MECHANT	MECHANT
gène 9	GENTIL	GENTIL	MECHANT	MECHANT
gène 10	GENTIL	GENTIL	MECHANT	MECHANT
gène 11	GENTIL	GENTIL	MECHANT	MECHANT
gène 12	GENTIL	GENTIL	MECHANT	MECHANT
gène 13	GENTIL	GENTIL	MECHANT	MECHANT
gène 14	GENTIL	GENTIL	MECHANT	MECHANT
gène 15	GENTIL	GENTIL	MECHANT	MECHANT
gène 16	GENTIL	GENTIL	MECHANT	MECHANT
gène 17	GENTIL	GENTIL	MECHANT	MECHANT
gène 18	GENTIL	GENTIL	MECHANT	MECHANT
gène 19	GENTIL	GENTIL	MECHANT	MECHANT
gène 20	GENTIL	GENTIL	MECHANT	MECHANT
gène 21	GENTIL	MECHANT	MECHANT	MECHANT
gène 22	GENTIL	MECHANT	MECHANT	MECHANT
gène 23	GENTIL	MECHANT	MECHANT	MECHANT
gène 24	GENTIL	MECHANT	MECHANT	MECHANT
gène 25	GENTIL	MECHANT	MECHANT	MECHANT
gène 26	GENTIL	MECHANT	MECHANT	MECHANT
gène 27	GENTIL	MECHANT	MECHANT	MECHANT
gène 28	GENTIL	MECHANT	MECHANT	MECHANT
gène 29	GENTIL	MECHANT	MECHANT	MECHANT
gène 30	GENTIL	MECHANT	MECHANT	MECHANT
gène 31	GENTIL	MECHANT	MECHANT	MECHANT
gène 32	GENTIL	MECHANT	MECHANT	MECHANT
gène 33	GENTIL	MECHANT	MECHANT	MECHANT
gène 34	GENTIL	MECHANT	MECHANT	MECHANT

gène 35	GENTIL	MECHANT	MECHANT	MECHANT
gène 36	GENTIL	MECHANT	MECHANT	MECHANT
gène 37	GENTIL	MECHANT	MECHANT	MECHANT
gène 38	GENTIL	MECHANT	MECHANT	MECHANT
gène 39	GENTIL	MECHANT	MECHANT	MECHANT
gène 40	GENTIL	MECHANT	MECHANT	MECHANT
gène 41	GENTIL	MECHANT	MECHANT	MECHANT
gène 42	GENTIL	MECHANT	MECHANT	MECHANT
gène 43	GENTIL	MECHANT	MECHANT	MECHANT
gène 44	GENTIL	MECHANT	MECHANT	MECHANT
gène 45	GENTIL	MECHANT	MECHANT	MECHANT
gène 46	GENTIL	MECHANT	MECHANT	MECHANT
gène 47	GENTIL	MECHANT	MECHANT	MECHANT
gène 48	GENTIL	MECHANT	MECHANT	MECHANT
gène 49	GENTIL	MECHANT	MECHANT	MECHANT
gène 50	GENTIL	MECHANT	MECHANT	MECHANT

environnement :les stratégies ne prenant jamais l'initiative de trahir

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	GENTIL	GENTIL	GENTIL	MECHANT
gène 2	GENTIL	GENTIL	GENTIL	MECHANT
gène 3	GENTIL	GENTIL	GENTIL	MECHANT
gène 4	GENTIL	GENTIL	GENTIL	MECHANT
gène 5	GENTIL	GENTIL	GENTIL	MECHANT
gène 6	GENTIL	GENTIL	GENTIL	MECHANT
gène 7	GENTIL	GENTIL	GENTIL	MECHANT
gène 8	GENTIL	GENTIL	GENTIL	MECHANT
gène 9	GENTIL	GENTIL	GENTIL	MECHANT
gène 10	GENTIL	GENTIL	GENTIL	MECHANT
gène 11	GENTIL	GENTIL	GENTIL	MECHANT
gène 12	GENTIL	GENTIL	GENTIL	MECHANT
gène 13	GENTIL	GENTIL	GENTIL	MECHANT
gène 14	GENTIL	GENTIL	GENTIL	MECHANT
gène 15	GENTIL	GENTIL	GENTIL	MECHANT
gène 16	GENTIL	GENTIL	GENTIL	MECHANT
gène 17	GENTIL	GENTIL	GENTIL	MECHANT
gène 18	GENTIL	GENTIL	GENTIL	MECHANT
gène 19	GENTIL	GENTIL	GENTIL	MECHANT
gène 20	GENTIL	GENTIL	GENTIL	MECHANT
gène 21	GENTIL	GENTIL	GENTIL	MECHANT
gène 22	GENTIL	GENTIL	GENTIL	MECHANT
gène 23	GENTIL	GENTIL	GENTIL	MECHANT
gène 24	GENTIL	GENTIL	GENTIL	MECHANT
gène 25	GENTIL	GENTIL	GENTIL	MECHANT
gène 26	GENTIL	GENTIL	GENTIL	MECHANT
gène 27	GENTIL	GENTIL	GENTIL	MECHANT
gène 28	GENTIL	GENTIL	GENTIL	MECHANT
gène 29	GENTIL	GENTIL	GENTIL	MECHANT
gène 30	GENTIL	GENTIL	GENTIL	MECHANT
gène 31	GENTIL	GENTIL	GENTIL	MECHANT
gène 32	GENTIL	GENTIL	GENTIL	MECHANT
gène 33	GENTIL	GENTIL	GENTIL	MECHANT

gène 34	GENTIL	GENTIL	MECHANT	MECHANT
gène 35	GENTIL	GENTIL	MECHANT	MECHANT
gène 36	GENTIL	GENTIL	MECHANT	MECHANT
gène 37	GENTIL	GENTIL	MECHANT	MECHANT
gène 38	GENTIL	GENTIL	MECHANT	MECHANT
gène 39	GENTIL	GENTIL	MECHANT	MECHANT
gène 40	GENTIL	GENTIL	MECHANT	MECHANT
gène 41	GENTIL	GENTIL	MECHANT	MECHANT
gène 42	GENTIL	GENTIL	MECHANT	MECHANT
gène 43	GENTIL	GENTIL	MECHANT	MECHANT
gène 44	GENTIL	GENTIL	MECHANT	MECHANT
gène 45	GENTIL	GENTIL	MECHANT	MECHANT
gène 46	GENTIL	GENTIL	MECHANT	MECHANT
gène 47	GENTIL	GENTIL	MECHANT	MECHANT
gène 48	GENTIL	GENTIL	MECHANT	MECHANT
gène 49	GENTIL	GENTIL	MECHANT	MECHANT
gène 50	GENTIL	GENTIL	MECHANT	MECHANT

environnement :les stratégies à jeux indépendants de celui de l'adversaire

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	MECHANT	MECHANT	MECHANT	MECHANT
gène 2	MECHANT	MECHANT	MECHANT	MECHANT
gène 3	MECHANT	MECHANT	MECHANT	MECHANT
gène 4	MECHANT	MECHANT	MECHANT	MECHANT
gène 5	MECHANT	MECHANT	MECHANT	MECHANT
gène 6	MECHANT	MECHANT	MECHANT	MECHANT
gène 7	MECHANT	MECHANT	MECHANT	MECHANT
gène 8	MECHANT	MECHANT	MECHANT	MECHANT
gène 9	MECHANT	MECHANT	MECHANT	MECHANT
gène 10	MECHANT	MECHANT	MECHANT	MECHANT
gène 11	MECHANT	MECHANT	MECHANT	MECHANT
gène 12	MECHANT	MECHANT	MECHANT	MECHANT
gène 13	MECHANT	MECHANT	MECHANT	MECHANT
gène 14	MECHANT	MECHANT	MECHANT	MECHANT
gène 15	MECHANT	MECHANT	MECHANT	MECHANT
gène 16	MECHANT	MECHANT	MECHANT	MECHANT
gène 17	MECHANT	MECHANT	MECHANT	MECHANT
gène 18	MECHANT	MECHANT	MECHANT	MECHANT
gène 19	MECHANT	MECHANT	MECHANT	MECHANT
gène 20	MECHANT	MECHANT	MECHANT	MECHANT
gène 21	MECHANT	MECHANT	MECHANT	MECHANT
gène 22	MECHANT	MECHANT	MECHANT	MECHANT
gène 23	MECHANT	MECHANT	MECHANT	MECHANT
gène 24	MECHANT	MECHANT	MECHANT	MECHANT
gène 25	MECHANT	MECHANT	MECHANT	MECHANT
gène 26	MECHANT	MECHANT	MECHANT	MECHANT
gène 27	MECHANT	MECHANT	MECHANT	MECHANT
gène 28	MECHANT	MECHANT	MECHANT	MECHANT
gène 29	MECHANT	MECHANT	MECHANT	MECHANT
gène 30	MECHANT	MECHANT	MECHANT	MECHANT
gène 31	MECHANT	GENTIL	MECHANT	MECHANT
gène 32	MECHANT	MECHANT	MECHANT	MECHANT

gène 33	MECHANT	MECHANT	MECHANT	MECHANT
gène 34	MECHANT	MECHANT	MECHANT	MECHANT
gène 35	MECHANT	MECHANT	MECHANT	MECHANT
gène 36	MECHANT	MECHANT	MECHANT	MECHANT
gène 37	MECHANT	MECHANT	MECHANT	MECHANT
gène 38	MECHANT	MECHANT	MECHANT	MECHANT
gène 39	MECHANT	MECHANT	MECHANT	MECHANT
gène 40	MECHANT	MECHANT	MECHANT	MECHANT
gène 41	MECHANT	MECHANT	MECHANT	MECHANT
gène 42	MECHANT	MECHANT	MECHANT	MECHANT
gène 43	MECHANT	MECHANT	MECHANT	MECHANT
gène 44	MECHANT	MECHANT	MECHANT	MECHANT
gène 45	MECHANT	MECHANT	MECHANT	MECHANT
gène 46	MECHANT	MECHANT	MECHANT	MECHANT
gène 47	MECHANT	MECHANT	MECHANT	MECHANT
gène 48	MECHANT	MECHANT	MECHANT	MECHANT
gène 49	MECHANT	MECHANT	MECHANT	MECHANT
gène 50	MECHANT	MECHANT	MECHANT	MECHANT

environnement :les stratégies à jeux dépendants de celui de l'adversaire

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	GENTIL	GENTIL	GENTIL	GENTIL
gène 2	GENTIL	GENTIL	GENTIL	GENTIL
gène 3	GENTIL	GENTIL	GENTIL	GENTIL
gène 4	GENTIL	GENTIL	GENTIL	GENTIL
gène 5	GENTIL	GENTIL	GENTIL	GENTIL
gène 6	GENTIL	GENTIL	GENTIL	GENTIL
gène 7	GENTIL	GENTIL	GENTIL	GENTIL
gène 8	GENTIL	GENTIL	GENTIL	GENTIL
gène 9	GENTIL	GENTIL	GENTIL	GENTIL
gène 10	GENTIL	GENTIL	GENTIL	GENTIL
gène 11	GENTIL	GENTIL	GENTIL	GENTIL
gène 12	GENTIL	GENTIL	GENTIL	MECHANT
gène 13	GENTIL	GENTIL	GENTIL	MECHANT
gène 14	GENTIL	GENTIL	GENTIL	MECHANT
gène 15	GENTIL	GENTIL	GENTIL	MECHANT
gène 16	GENTIL	GENTIL	GENTIL	MECHANT
gène 17	GENTIL	GENTIL	GENTIL	MECHANT
gène 18	GENTIL	GENTIL	GENTIL	MECHANT
gène 19	GENTIL	GENTIL	GENTIL	MECHANT
gène 20	GENTIL	GENTIL	GENTIL	MECHANT
gène 21	GENTIL	GENTIL	GENTIL	MECHANT
gène 22	GENTIL	GENTIL	GENTIL	MECHANT
gène 23	GENTIL	GENTIL	GENTIL	MECHANT
gène 24	GENTIL	GENTIL	GENTIL	MECHANT
gène 25	GENTIL	GENTIL	GENTIL	MECHANT
gène 26	GENTIL	GENTIL	GENTIL	MECHANT
gène 27	GENTIL	GENTIL	GENTIL	MECHANT
gène 28	GENTIL	GENTIL	GENTIL	MECHANT
gène 29	GENTIL	GENTIL	GENTIL	MECHANT
gène 30	GENTIL	GENTIL	GENTIL	MECHANT
gène 31	GENTIL	GENTIL	GENTIL	MECHANT

gène 32	GENTIL	GENTIL	GENTIL	MECHANT
gène 33	GENTIL	GENTIL	GENTIL	MECHANT
gène 34	GENTIL	GENTIL	GENTIL	MECHANT
gène 35	GENTIL	GENTIL	GENTIL	MECHANT
gène 36	GENTIL	GENTIL	GENTIL	MECHANT
gène 37	GENTIL	GENTIL	GENTIL	MECHANT
gène 38	GENTIL	GENTIL	GENTIL	MECHANT
gène 39	GENTIL	GENTIL	GENTIL	MECHANT
gène 40	GENTIL	GENTIL	GENTIL	MECHANT
gène 41	GENTIL	GENTIL	GENTIL	MECHANT
gène 42	GENTIL	GENTIL	GENTIL	MECHANT
gène 43	GENTIL	GENTIL	GENTIL	MECHANT
gène 44	GENTIL	GENTIL	GENTIL	MECHANT
gène 45	GENTIL	GENTIL	GENTIL	MECHANT
gène 46	GENTIL	GENTIL	GENTIL	MECHANT
gène 47	GENTIL	GENTIL	GENTIL	MECHANT
gène 48	GENTIL	GENTIL	GENTIL	MECHANT
gène 49	GENTIL	GENTIL	GENTIL	MECHANT
gène 50	GENTIL	GENTIL	GENTIL	MECHANT

environnement :les stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux

	50 premiers	**100 premiers**	**150 premiers**	**200 premiers**
gène 1	GENTIL	GENTIL	GENTIL	MECHANT
gène 2	GENTIL	GENTIL	MECHANT	MECHANT
gène 3	GENTIL	GENTIL	MECHANT	MECHANT
gène 4	GENTIL	GENTIL	MECHANT	MECHANT
gène 5	GENTIL	GENTIL	MECHANT	MECHANT
gène 6	GENTIL	GENTIL	MECHANT	MECHANT
gène 7	GENTIL	GENTIL	MECHANT	MECHANT
gène 8	GENTIL	GENTIL	MECHANT	MECHANT
gène 9	GENTIL	GENTIL	MECHANT	MECHANT
gène 10	GENTIL	GENTIL	MECHANT	MECHANT
gène 11	GENTIL	GENTIL	MECHANT	MECHANT
gène 12	GENTIL	GENTIL	MECHANT	MECHANT
gène 13	GENTIL	GENTIL	MECHANT	MECHANT
gène 14	GENTIL	GENTIL	MECHANT	MECHANT
gène 15	GENTIL	GENTIL	MECHANT	MECHANT
gène 16	GENTIL	GENTIL	MECHANT	MECHANT
gène 17	GENTIL	GENTIL	MECHANT	MECHANT
gène 18	GENTIL	GENTIL	MECHANT	MECHANT
gène 19	GENTIL	GENTIL	MECHANT	MECHANT
gène 20	GENTIL	GENTIL	MECHANT	MECHANT
gène 21	GENTIL	GENTIL	MECHANT	MECHANT
gène 22	GENTIL	GENTIL	MECHANT	MECHANT
gène 23	GENTIL	GENTIL	MECHANT	MECHANT
gène 24	GENTIL	GENTIL	MECHANT	MECHANT
gène 25	GENTIL	GENTIL	MECHANT	MECHANT
gène 26	GENTIL	GENTIL	MECHANT	MECHANT
gène 27	GENTIL	GENTIL	MECHANT	MECHANT
gène 28	GENTIL	GENTIL	MECHANT	MECHANT
gène 29	GENTIL	GENTIL	MECHANT	MECHANT

gène 30	GENTIL	GENTIL	MECHANT	MECHANT
gène 31	GENTIL	GENTIL	MECHANT	MECHANT
gène 32	GENTIL	GENTIL	MECHANT	MECHANT
gène 33	GENTIL	GENTIL	MECHANT	MECHANT
gène 34	GENTIL	GENTIL	MECHANT	MECHANT
gène 35	GENTIL	GENTIL	MECHANT	MECHANT
gène 36	GENTIL	GENTIL	MECHANT	MECHANT
gène 37	GENTIL	GENTIL	MECHANT	MECHANT
gène 38	GENTIL	GENTIL	MECHANT	MECHANT
gène 39	GENTIL	GENTIL	MECHANT	MECHANT
gène 40	GENTIL	GENTIL	MECHANT	MECHANT
gène 41	GENTIL	GENTIL	MECHANT	MECHANT
gène 42	GENTIL	GENTIL	MECHANT	MECHANT
gène 43	GENTIL	GENTIL	MECHANT	MECHANT
gène 44	GENTIL	GENTIL	MECHANT	MECHANT
gène 45	GENTIL	GENTIL	MECHANT	MECHANT
gène 46	GENTIL	GENTIL	MECHANT	MECHANT
gène 47	GENTIL	GENTIL	MECHANT	MECHANT
gène 48	GENTIL	GENTIL	MECHANT	MECHANT
gène 49	GENTIL	GENTIL	MECHANT	MECHANT
gène 50	GENTIL	GENTIL	MECHANT	MECHANT

Annexe 3 : pattern génétique des stratégies d'affrontement ayant gagné dans chacun la confrontation écologique dans chacun de nos environnements stratégiques

	10 coups	**20 coups**	**30 coups**	**40 coups**	**50 coups**	**100 coups**	**100 coups (suite)**
gène 1	PROB_CO	TESTCON	PROB_ME	PROB_CO	GENTIL	PAV_CIN	T_F_T_DU
gène 2	PROB_ME	PROB_ME	PROB_ME	PROB_CO	GENTIL	PROB_ME	T_F_T_DU
gène 3	PROB_ME	PROB_ME	PROB_ME	PROB_CO	GENTIL	PROB_ME	T_F_T_DU
gène 4	PROB_CO	TESTCON	PROB_CO	PROB_CO	GENTIL	PROB_ME	T_F_T_DU
gène 5	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	GENTIL	PROB_ME	T_F_T_DU
gène 6	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	GENTIL	PROB_CO	T_F_T_DU
gène 7	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 8	MECHANT	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 9	MECHANT	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 10	MECHANT	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 11		T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 12		T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 13		T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 14		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 15		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 16		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 17		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 18		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 19		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 20		RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 21			T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 22			T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 23			T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 24			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 25			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 26			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 27			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 28			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE

gène 29			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 30			RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 31				T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 32				T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE
gène 33				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 34				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 35				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 36				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 37				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 38				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 39				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 40				T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE
gène 41					RANCUNIE	T_F_T_DU	RANCUNIE
gène 42					RANCUNIE	T_F_T_DU	RANCUNIE
gène 43					RANCUNIE	T_F_T_DU	RANCUNIE
gène 44					RANCUNIE	T_F_T_DU	RANCUNIE
gène 45					RANCUNIE	T_F_T_DU	RANCUNIE
gène 46					RANCUNIE	T_F_T_DU	RANCUNIE
gène 47					RANCUNIE	T_F_T_DU	RANCUNIE
gène 48					RANCUNIE	T_F_T_DU	RANCUNIE
gène 49					RANCUNIE	T_F_T_DU	RANCUNIE
gène 50					RANCUNIE	T_F_T_DU	RANCUNIE

Annexe 4 : stratégie d'affrontement gagnant les confrontations écologiques avec chacun de nos environnements stratégiques.

	200 coups	200 coups (suite)	200 coups (suite)	200 coups (suite)
gène 1	PAV_CIN	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 2	PAV_CIN	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 3	PAV_CIN	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 4	PAV_CIN	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 5	PAV_CIN	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 6	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 7	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 8	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 9	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 10	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 11	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 12	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 13	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 14	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 15	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 16	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 17	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 18	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 19	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 20	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 21	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 22	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 23	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 24	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 25	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 26	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 27	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 28	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 29	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 30	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 31	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 32	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU

gène 33	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 34	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 35	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 36	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 37	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 38	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 39	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 40	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 41	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 42	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 43	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 44	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 45	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 46	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 47	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 48	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 49	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU
gène 50	T_F_T_DU	T_F_T_DU	T_F_T_DU	PERIOD_MECH

Annexe 5 : Stratégies formées lors d'expériences dans l'environnement de Delahaye et Mathieu

	moyenne 1	meilleure 1	moyenne 2	meilleure 2	moyenne 3	meilleure 3	moyenne 4	meilleure 4
gène 1	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 2	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 3	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 4	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 5	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 6	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 7	MAJ_CIN	PROB_CO	PROB_ME	MAJ_CIN	PROB_ME	GENTIL	PAV_CIN	TESTCON
gène 8	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PAV_CIN	TESTCON
gène 9	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PAV_CIN	TESTCON
gène 10	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 11	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 12	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 13	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 14	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	GENTIL	PROB_ME	TESTCON
gène 15	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 16	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 17	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 18	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 19	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 20	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 21	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 22	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 23	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 24	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 25	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	TESTCON
gène 26	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 27	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 28	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 29	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 30	MAJ_CIN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 31	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 32	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL

gène 33	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 34	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 35	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 36	SEUILREN	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 37	T_F_T_DU	PROB_CO	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 38	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	T_F_T_DU	PROB_ME	SEUIL
gène 39	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 40	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 41	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 42	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 43	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 44	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 45	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 46	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 47	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 48	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 49	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 50	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 51	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 52	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 53	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 54	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 55	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	PROB_ME	SEUIL
gène 56	T_F_T_DU	T_F_T_DU	PROB_ME	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 57	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 58	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 59	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 60	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 61	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 62	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 63	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 64	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 65	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 66	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 67	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 68	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 69	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 70	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 71	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 72	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 73	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 74	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 75	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 76	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 77	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 78	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 79	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 80	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 81	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 82	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 83	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 84	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 85	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	TIT_FOR_T	T_F_T_DU	RANCUNIE	RANCUNIE
gène 86	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	TIT_FOR_T	T_F_T_DU	RANCUNIE	RANCUNIE

gène 87	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	TIT_FOR_T	T_F_T_DU	RANCUNIE	RANCUNIE
gène 88	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	TIT_FOR_T	T_F_T_DU	RANCUNIE	RANCUNIE
gène 89	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE	RANCUNIE
gène 90	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE	RANCUNIE
gène 91	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE	RANCUNIE
gène 92	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE	RANCUNIE
gène 93	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	T_F_T_DU	RANCUNIE	RANCUNIE
gène 94	RANCUNIE	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 95	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 96	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 97	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 98	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 99	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE
gène 100	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE	T_F_T_DU	T_F_T_DU	RANCUNIE	RANCUNIE

Annexe 6 : résultats des stratégies de l'**Annexe 5** lors de confrontations écologiques face à tous les environnements : (env 1 = environnement composé de toutes les stratégies, env 2 = environnement composé des stratégies pouvant pRENdre l'initiative de la trahison, env 3 = environnement composé des stratégies ne pRENant jamais l'initiative de la trahison, env 4 = environnement composé des stratégies à jeux indépendants de celui de l'adversaire, env 5 = environnement composé de stratégies à jeux dépendants de celui de l'adversaire, env 6 = environnement composé des stratégies à jeux dépendants de la réponse de l'adversaire à leurs jeux, env 7 = environnement de Delahaye et Mathieu)

	moyenne 1	**meilleure1**	**moyenne 2**	**meilleure 2**	**moyenne 3**	**meilleure 3**	**moyenne 4**	**meilleure 4**
env 1	G	G	G	G	G	G	G	G
env 2	G	G	G	G	G	G	G	G
env 3	G	G	G	G	G	G	G	G
env 4	G	G	G	G	G	G	G	G
env 5	G	G	G	G	G	G	G	S
env 6	G	G	G	G	G	G	G	G
env 7	G	G	G	G	G	G	G	G

Annexe 7 : évolution du score moyen des meilleures stratégies en fonction du nombre de crossing-overs ; les paramètres étant 100 coups, 25 croisements et 8 reproductions par croisement et 50 tests.

	nombre impair	**nombre pair**
1 crossing-over	5503	
2 crossing-over		5328
3 crossing-over	5438	
4 crossing-over		5268
5 crossing-over	5370	
6 crossing-over		5235
7 crossing-over	5287	
8 crossing-over		5209
9 crossing-over	5248	
10 crossing-over		5172

Annexe 8 : évolution du score moyen des meilleures stratégies en fonction du nombre de crossing-overs ; les paramètres étant 200 coups, 25 croisements et 8 reproductions par croisement et 50 tests.

	nombre impair	nombre pair
1 crossing-over	10977	
2 crossing-over		10679
3 crossing-over	10847	
4 crossing-over		10592
5 crossing-over	10682	
6 crossing-over		10442
7 crossing-over	10498	
8 crossing-over		10406
9 crossing-over	10424	
10 crossing-over		10332

Annexe 9 : Proportions des stratégies (gènes) GENTIL et MECHANT dans les stratégies au fur et à mesure que le nombre de gènes (coups) augmente dans ces stratégies. Les paramètres sont : 100 croisements, 8 reproductions par croisement, 1 crossing-over par reproduction et 50 tests.

environnement stratégique composé de toutes les stratégies

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0,5	0,5
pour 3 gènes :	0,6666667	0,3333333
pour 4 gènes :	0,5	0,5
pour 5 gènes :	0,416	0,584
pour 6 gènes :	0,5033333	0,4966667
pour 7 gènes :	0,5685714	0,4314286
pour 8 gènes :	0,5325	0,4675
pour 9 gènes :	0,5755556	0,4244444
pour 10 gènes :	0,53	0,47
pour 11 gènes :	0,5490909	0,4509091
pour 12 gènes :	0,5283333	0,4716667
pour 13 gènes :	0,5630769	0,4369231
pour 14 gènes :	0,51	0,49
pour 15 gènes :	0,528	0,472
pour 100 gènes	0,5264	0,4736
pour 200 gènes	0,5317	0,4683

environnement stratégique composé des stratégies pouvant pRENdre l'initiative de trahir

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0	1
pour 3 gènes :	0	1
pour 4 gènes :	0,08	0,92
pour 5 gènes :	0,072	0,928
pour 6 gènes :	0,17	0,83
pour 7 gènes :	0,2257143	0,7742857
pour 8 gènes :	0,3475	0,6525
pour 9 gènes :	0,3022222	0,6977778
pour 10 gènes :	0,284	0,716
pour 11 gènes :	0,3272727	0,6727273
pour 12 gènes :	0,3716667	0,6283333
pour 13 gènes :	0,3369231	0,6630769
pour 14 gènes :	0,3142857	0,6857143
pour 15 gènes :	0,3546667	0,6453333
pour 200 gènes	0,4095	0,5905

environnement stratégique composé des stratégies ne pRENant jamais l'initiative de la trahison

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0,5	0,5
pour 3 gènes :	0,6666667	0,3333333
pour 4 gènes :	0,5	0,5
pour 5 gènes :	0,6	0,4
pour 6 gènes :	0,51	0,49
pour 7 gènes :	0,5714286	0,4285714
pour 8 gènes :	0,6225	0,3775
pour 9 gènes :	0,6644444	0,3355556
pour 10 gènes :	0,696	0,304
pour 11 gènes :	0,7254545	0,2745455
pour 12 gènes :	0,74	0,26
pour 13 gènes :	0,6707692	0,3292308
pour 14 gènes :	0,7142857	0,2857143
pour 15 gènes :	0,736	0,264
pour 100 gènes	0,6682	0,3318
pour 200 gènes	0,6672	0,3328

environnement stratégique composé des stratégies à jeux indépendants de celui de l'adversaire

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0	1
pour 3 gènes :	0	1
pour 4 gènes :	0	1
pour 5 gènes :	0	1
pour 6 gènes :	0	1
pour 7 gènes :	0	1
pour 8 gènes :	0	1
pour 9 gènes :	0	1
pour 10 gènes :	0	1
pour 11 gènes :	0	1
pour 12 gènes :	0	1
pour 13 gènes :	0	1
pour 14 gènes :	0	1
pour 15 gènes :	0	1
pour 200 gènes	0,0005	0,9995

environnement stratégique composé des stratégies à jeux dépendants de celui de l'adversaire

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0,5	0,5
pour 3 gènes :	0,6666667	0,3333333
pour 4 gènes :	0,5	0,5
pour 5 gènes :	0,6	0,4
pour 6 gènes :	0,5033333	0,4966667
pour 7 gènes :	0,6057143	0,3942857
pour 8 gènes :	0,6225	0,3775
pour 9 gènes :	0,6666667	0,3333333
pour 10 gènes :	0,694	0,306
pour 11 gènes :	0,7272727	0,2727273
pour 12 gènes :	0,75	0,25
pour 13 gènes :	0,7692308	0,2307692
pour 14 gènes :	0,7857143	0,2142857
pour 15 gènes :	0,8	0,2
pour 100 gènes	0,8176	0,1824
pour 200 gènes	0,8159	0,1841
pour 500 gènes	0,8038	0,1962

environnement stratégique composé des stratégies à jeux dépendants de la réponse de l'adversaire à leur propre jeu

	GENTIL	MECHANT
pour 1 gène :	0	1
pour 2 gènes :	0	1
pour 3 gènes :	0	1
pour 4 gènes :	0,25	0,75
pour 5 gènes :	0,008	0,992
pour 6 gènes :	0,5	0,5
pour 7 gènes :	0,5714286	0,4285714
pour 8 gènes :	0,56	0,44
pour 9 gènes :	0,5933333	0,4066667
pour 10 gènes :	0,512	0,488
pour 11 gènes :	0,5618182	0,4381818
pour 12 gènes :	0,5483333	0,4516667
pour 13 gènes :	0,5523077	0,4476923
pour 14 gènes :	0,5128571	0,4871429
pour 15 gènes :	0,5333333	0,4666667
pour 200 gènes	0,5164	0,4836
pour 500 gènes	0,5082	0,4918

Le code source

Pour des raisons de clarté, nous avons mis en gras les commentaires, et nous n'avons pas jugé important d'insérer le code correspondant aux positionnements des objets interactifs concrets.

Fichier contenant les fonctions et les variables

```
# include <iostream.h>
# include <string.h>
# include <stdlib.h>
# include <conio.h>
# include <stdio.h>
# include <math.h>

#include "Strate1FP.h"
#include "Strate2FP.h"
#include "Strate3FP.h"
#include "Strate4FP.h"
#include "Strate5FP.h"


#define tc 5
#define cc 3
#define rr 2
#define tt 1
#define ct 0
#define NombreStrategiesDifferentes 20
#define EffectifsConstantsTestees 100
#define EffectifLimite 99


//le déroulement du programme correspond à une évolution des effectifs des stratégies testées dans un environnement particulier et fixe composé de stratégies représentatives ; il est important de remarquer que nous employons le terme de stratégies représentatives comme synonyme de stratégies environnementales

int c,i,NumTest,NombreParts,NombreGenerations,NombreCroisements,NombreCrossingOver,
NombreTesteesDepart;

//NombreParts correspond au nombre de gènes de chaque chromosome et qui est donc identique pour chaque stratégie
//NombreGenerations correspond au nombre de générations que l'on veut développer avant un croisement (phase où certaines stratégies se reproduisent de façon sexuée) ; une génération est développée quand les effectifs des stratégies testées ont tous obtenus leur score face aux effectifs des représentatives et que les nouveaux effectifs des stratégies testées ont été calculés
//NombreCroisements c'est le nombre de fois qu'il y a reproduction sexuée lors d'une évolution
//NombreCrossingOver c'est le nombre de crossing-over
//EffectifsNouvellesStrategies ce sont les effectifs des nouvelles stratégies apparues suite aux crossing-over

 int NombreStrategiesTestees,NombreStrategiesRepresentatives, EffectifsTotauxTestees,
NombreStrategiesApparaissants;
 struct TGene              // structure d'un gène, chaque chromosome représentant une stratégie est constitué d'une suite de gènes
 {
        short int Strategie ;     // la stratégie est le nom de la stratégie c'est en fait un entier
```

```
        struct TGene * Suivant;
};
struct TJeu          //un jeu joué par un gène, utilisé par AFFRONTEMENT pour constituer une liste de jeux joués par une stratégie (utile pour les fonctions qui ont un jeu non déterminé)
{
 char Coup;          //c'est 0 si on a trahis et 1 si on a coopéré
 struct TJeu * Suivant;
} ;


struct TGene * ListeStrategiesTestees [10000];
//tableaux contenant les pointeurs vers les stratégies à tester; ces stratégies feront l'objet de crossing-over
struct TGene * ListeStrategiesRepresentatives [100];
//tableaux contenant les pointeurs vers les stratégies représentatives (maximum 100 ici) et qui ne feront pas l'objet de crossing-over

int ScoreStrategiesTestees [10000];   //un tableau avec les scores de stratégies testées
double EffectifsStrategiesTestees[10000];   //un tableau pour les effectifs des stratégies
int EffectifsStrategiesRepresentatives[100], NouvellesStrategies[10000], TotauxGenes[100];  //un tableau contenant le total des différents gènes des meilleures stratégies
int StrategiesACroiser[10000];    //un tableau contenant NombreStrategiesApparaissant numéros correspondant aux stratégies à croiser

********FONCTION POUR CORRESPONDANCE DES GENES AVEC LES NUMEROS************

AnsiString CORRESPONDANCE(short int NumStrat)
{
switch (NumStrat){
 case 1: return "GENTIL";
      break;
 case 2: return "MECHANT";
      break;
 case 3: return "LUNATIQUE";
      break;
 case 4: return "TIT_FOR_TAT";
      break;
 case 5: return "RANCUNIER";
      break;
 case 6: return "PERIOD_MECH";
      break;
 case 7: return "PERIOD_GENT";
      break;
 case 8: return "MAJ_MOU";
      break;
 case 9: return "SEUIL";
      break;
 case 10: return "BINAIRE";
      break;
 case 11: return "MAJ_DUR";
      break;
 case 12: return "SEUILREN";
      break;
 case 13: return "MEFIANT";
      break;
 case 14: return "T_F_T_DUR";
      break;
 case 15: return "MAJ_CINQ";
      break;
 case 16: return "TESTCONTINUE";
      break;
```

```
  case 17: return "PAVLOV";
       break;
  case 18: return "PAV_CINQ";
       break;
  case 19: return "PROB_MEME_QUE";
       break;
  case 20: return "PROB_CONTINUER";
       break;
  default: return " ";
 }
}
```

******FONCTION POUR LE MARQUAGE DES GENES DES STRATEGIES DE DEPART************

//crée les listes de stratégies respectivement représentatives et à tester et appelle REMPLIR_STRATEGIE qui crée un chromosome dont le nombre de gènes est égal au nombre de parts ou de coups à jouer dans une partie et qui contiennent le numéro permettant de les situer sur le chromosome ainsi que le nom de la stratégie de départ à laquelle il appartiennent. Parallèllement un tableau contenant les effectifs des stratégies est créé.

```
REMPLIR_STRATEGIE(short int NomStrategie , struct TGene * PGene)  //reçoit l'adresse du premier gène
préalablement créé et le nom de la stratégie et s'occupe de créer un chromosome dont les gènes portent le
nom de la stratégie et le nombre de gènes est égal au nombre de parts jouées
{
 int i;
 struct TGene * aLocPGene;      //initialisation de deux pointeurs locaux vers une structure TGene
 struct TGene * bLocPGene;

 (* PGene).Strategie= NomStrategie;  //une fonction d'assignation d'un string
 if (NombreParts==1)          // si il n'y a qu'un gène à remplir, alors le pointeur vers le suivant est mis à
nul
        (* PGene).Suivant=NULL;

 aLocPGene=PGene;             // un pointeur local prend l'adresse du pointeur vers le premier gène pour
effectuer les opérations

 for (i=2; i<NombreParts+1; i++) // pour i allant de 2 au nombre de parts jouées on rempli les
caractéristiques des gènes
 {
        bLocPGene=new struct TGene; // on crée le gène qui suivra et ensuite on introduit ses
caractéristiques
  (* bLocPGene).Strategie= NomStrategie;
        if (i==NombreParts)
                (* bLocPGene).Suivant= NULL; //on libère la mémoire vers laquelle pointe le pointeur du
dernier gène
        (* aLocPGene).Suivant=bLocPGene;
        aLocPGene=(* aLocPGene).Suivant;
 }
}
```

JEUX JOUES PAR UN GENE SELON SON APPARTENANCE A UNE STRATEGIE (FONCTIONS)

```
REMPLIR_LISTES()
{
 short int i;
 int c,compteur;
 NombreStrategiesTestees=0;
 NombreStrategiesRepresentatives=0;
 for (i=1; i<(*(*Form1).StringGrid1).RowCount; i++)
```

```
 {
  if (StrToInt((*(*Form1).StringGrid1).Cells[1][i])>0)
    NombreStrategiesTestees=NombreStrategiesTestees+StrToInt((*(*Form1).StringGrid1).Cells[1][i]);
  if (StrToInt((*(*Form1).StringGrid2).Cells[1][i])>0)

NombreStrategiesRepresentatives=NombreStrategiesRepresentatives+StrToInt((*(*Form1).StringGrid2).Cells[1
][i]);
 }
 NombreTesteesDepart=NombreStrategiesTestees;
 compteur=0;
 for (i=1; i<(*(*Form1).StringGrid1).RowCount; i++)
 {
    for (c=1; c<=StrToInt((*(*Form1).StringGrid1).Cells[1][i]); c++)
    {
     compteur++;
     ListeStrategiesTestees [compteur]=new struct TGene; // création du premier gène des stratégies à tester
de la liste remarquons que le tableau de pointeur est une variable globale pour jouer plus facilement avec
les stratégies
     REMPLIR_STRATEGIE(i, ListeStrategiesTestees [compteur]); //on passe l'adresse de la structure en
paramètre variable et le numéro de ligne qui correspond à une stratégie
    }
 }
 for (c=1; c<NombreStrategiesTestees+1; c++)
  EffectifsStrategiesTestees[c]=EffectifsConstantsTestees;

 compteur=0;
 for (i=1; i<(*(*Form1).StringGrid2).RowCount; i++)
 {
    for (c=1; c<=StrToInt((*(*Form1).StringGrid2).Cells[1][i]); c++)
    {
     compteur++;
     ListeStrategiesRepresentatives [compteur]=new struct TGene; // création du premier gène des stratégies
à tester de la liste remarquons que le tableau de pointeur est une variable globale pour jouer plus
facilement avec les stratégies
     REMPLIR_STRATEGIE(i, ListeStrategiesRepresentatives [compteur]); //on passe l'adresse de la
structure en paramètre variable
    }
 }
 for (c=1; c<NombreStrategiesRepresentatives+1; c++)
   EffectifsStrategiesRepresentatives[c]=EffectifsConstantsTestees;

}
```

JEUX JOUES PAR UN GENE SELON SON APPARTENANCE A UNE STRATEGIE (FONCTIONS)

```
//c pour coopérer, t pour trahir et r pour renoncer

char JEU_GENTILLE()
{
 return 'c';
}

char JEU_MECHANTE()
{
 return 't';
}

char JEU_LUNATIQUE()
{
 if ((rand()%2) == 0)
```

```
   return 'c';
   else return 't';
}

char JEU_TIT_FOR_TAT (struct TJeu * Jeu, int i)
{
 if (i==1)
   return 'c';
 else
 {
  if ((*Jeu).Coup == 't')
    return 't';
    else return 'c';
 }
}

char JEU_RANCUNIERE (struct TJeu * Jeu, int i)
{
 short int Trouve=0;
 struct TJeu * aPJeu;

 if (i ==1)
  return 'c';
 else
 {
  aPJeu=Jeu;
  while ((aPJeu!=NULL)&&!(Trouve))
  {
   if (((*aPJeu).Coup == 'c')||((*aPJeu).Coup == 'r'))
     aPJeu=(*aPJeu).Suivant;
   else Trouve=1;
  }
  if (Trouve==1)
    return 't';
  else return 'c';
 }
}

char JEU_PERIODIQUE_MECHANTE (short int NumPart)
{
 if (NumPart % 3 == 0) return 'c';
   else return 't';
}

char JEU_PERIODIQUE_GENTILLE (short int NumPart)
{
 if (NumPart % 3 == 0) return 't';
   else return 'c';
}

char JEU_MAJ_MOU( struct TJeu * Jeu, int i)
{
 int j, numbt=0, numbc=0;

 for (j=1; j<i; j++)
 {
  if (((*Jeu).Coup)=='c')
    numbc++;
  else if (((*Jeu).Coup)=='t' )
   numbt++;
```

```
   Jeu= (*Jeu).Suivant;
  }
  if (numbt > numbc)
    return 't';
  else return 'c';
}

char JEU_SEUIL(short int Resultat, short int NombreCoupsJoues)
{
  struct TJeu * aPJeu;
  float Seuil=2.5 ;

if (NombreCoupsJoues==1)
    return 'c';
  else
  {
  if (Resultat/(NombreCoupsJoues-1)<Seuil)
    return 't';
    else return 'c';
  }
}

char JEU_BINAIRE(int i)
{
 if (i % 2 == 0) return 't';
   else return 'c';
}

char JEU_MAJ_DUR( struct TJeu * Jeu, int i)
{
  int j, numbt=0, numbc=0;

  for (j=1; j<i; j++)
  {
   if (((*Jeu).Coup)=='c')
     numbc++;
   else if (((*Jeu).Coup)=='t' )
     numbt++;
   Jeu= (*Jeu).Suivant;
  }
  if (numbt >= numbc)
    return 't';
  else return 'c';
}

char JEU_SEUIL_AVEC_RENONCEMENT(short int Resultat, short int NombreCoupsJoues)
{
  struct TJeu * aPJeu;
  float SeuilRen, SeuilTrah ;

  if (NombreCoupsJoues==1)
    return 'c';
  else
  {
   SeuilRen=2;
   SeuilTrah=4;
    if (Resultat/(NombreCoupsJoues-1)<SeuilRen)
      return 'r';
    else if (Resultat/(NombreCoupsJoues-1)>SeuilTrah)
        return 't';
```

```
        else return 'c';
  }
}


char MEFIANT(struct TJeu * Jeu)   //elle trahit donc quand l'autre a joué un renoncement la fois d'avant
{
   if ((Jeu != NULL) && (*Jeu).Coup == 'c')
    return 'c';
    else return 't';
}

char TIT_FOR_TAT_DUR(struct TJeu * Jeu, int i)
{
 if (i==1)
  return 'c';
 else
 {
  if ((*Jeu).Coup == 'c')
    {
    if ((*Jeu).Suivant != NULL)
     {
     Jeu = (*Jeu).Suivant;
       if ((*Jeu).Coup == 'c')
         return 'c';
       else return 't';
     }
    else return 'c';
    }
  else return 't';
 }
}

char MAJ_CINQ(struct TJeu * Jeu,int i)
{
 int j=5;  //j pour la période prise en compte
 int numbt=0, numbc=0;

 if (i<j)
  j=i;
 while ( (j>1) )
  {
    if ((*Jeu).Coup == 't')
      numbt++;
    else numbc++;
    Jeu = (*Jeu).Suivant;
    j--;
  }
  if (numbt > numbc)
    return 't';
  else return 'c';
}

char CONTINUE (struct TJeu * Jeu)
{
   if ((Jeu != NULL)&&((*Jeu).Suivant != NULL))
     {
     if ( (*Jeu).Coup ==(*(*Jeu).Suivant).Coup)
       return 'c';
     else return 't';
```

```
    }
   else return 'c';
}

char PAVLOV (struct TJeu * toi, struct TJeu * moi)
{
if (moi == NULL)
 return 'c';
else
   if ((*toi).Coup == (*moi).Coup)
     return 'c';
   else return 't';
}


char PAVLOV_CINQ (struct TJeu * toi, struct TJeu * moi, int i)
{
int j=6,compttoi=0, comptmoi=0;

 if (i<j) j=1;
 while ( (j>1) )
 {
  if ((*toi).Coup=='t')
    compttoi++;
  else compttoi--;

  if ((*moi).Coup=='t')
    comptmoi++;
  else comptmoi--;

  toi = (*toi).Suivant;
  moi = (*moi).Suivant;
  j--;
 }

 if ( ( (compttoi<0)&& (comptmoi<0)) || ( (compttoi >= 0)&& (comptmoi >= 0)))
 return 'c';
else return 't';
}


char PROBA_MEME_QUE_LUI (struct TJeu * toi, struct TJeu * moi, int i)
{
int numbt=0, numbc=0;
char c1;

  if ( moi != NULL)
  {
   c1 = (*moi).Coup;
   moi = (*moi).Suivant;
   i--;
   while ( (moi != NULL) && (i>1) )
   {
    if ((*moi).Coup==c1)
    {
      if ((*toi).Coup=='c')
         numbc++;
      else numbt++;
    }
    moi = (*moi).Suivant;
```

```
    toi = (*toi).Suivant;
    i--;
   }
   if (numbc >= numbt)
     return 'c';
   else return 't';
   }
  else return 'c';
}


char PROBA_CONTINUE (struct TJeu * toi, struct TJeu * moi, int i)
{
int numbt=0, numbc=0;
char c1;

  if ( moi != NULL)
  {
   c1 = (*moi).Coup;
   moi = (*moi).Suivant;
   i--;
   while ( (moi != NULL) && (i>1) )
   {
    if ((*moi).Coup==c1)
    {
      if ((*toi).Coup=='c')
         numbc++;
      else numbt++;
    }
    moi = (*moi).Suivant;
    toi = (*toi).Suivant;
    i--;
   }
   if (numbc >= numbt)
     return c1;
   else
      {if (c1 == 'c')
        return 't';
       else return 'c';
      }
  }
  else return 'c';
}
```

****SUPPRIMER LES LISTES DES JEUX CONSTITUEES LORS DE CHAQUE AFFRONTEMENT*****

```
SUPPRIMER_JEUX(struct TJeu * ListeJeux)
{
 struct TJeu * PListeJeux;
 int i;

 for (i=1; i<NombreParts+1; i++)
 {
  if (i<NombreParts)
    PListeJeux= (*ListeJeux).Suivant;
  delete ListeJeux;
  ListeJeux = PListeJeux;
 }
```

```
}

*********AFFRONTEMENT ENTRE DEUX STRATEGIES*************


void AFFRONTEMENT( struct TGene* strat1 ,struct TGene* strat2, int *resultat1,int *resultat2 )
//reçoit les adresses d'une stratégie à tester et d'une représentative et stocke le résultat de l'affrontement à
deux adresses
{

        int i, res1, res2;
        short int xrep,yrep;            // 0 si trahir, 1 si coopérer ; c'est la réponse d'un gène lors d'un coup en
fonction de sa stratégie
        struct TJeu * aPJeuX, * aPJeuY,* aPListeJeuxX,* aPListeJeuxY ;
        short int temp; //reçoit le nom de stratégie d'origine du gène
  struct TGene * X, * Y;

        res1 = 0;
        res2 = 0;

 X=strat1;
 Y=strat2;

 aPListeJeuxX = NULL;

 aPListeJeuxY = NULL;

        for (i=1; i<= NombreParts ;i++)
        {
       temp = (*X).Strategie;
       switch ( temp ){

         case 1: xrep = JEU_GENTILLE();
               break;
         case 2: xrep = JEU_MECHANTE();
               break;
         case 3: xrep = JEU_LUNATIQUE();
               break;
         case 4: xrep = JEU_TIT_FOR_TAT( aPListeJeuxY, i);
               break;
         case 5: xrep = JEU_RANCUNIERE(aPListeJeuxY, i);
               break;
         case 6: xrep = JEU_PERIODIQUE_MECHANTE (i);
               break;
         case 7: xrep = JEU_PERIODIQUE_GENTILLE (i);
               break;
         case 8: xrep = JEU_MAJ_MOU (aPListeJeuxY, i);
               break;
         case 9: xrep = JEU_SEUIL(res1, i);
               break;
         case 10:xrep = JEU_BINAIRE(i);
               break;
         case 11:xrep = JEU_MAJ_DUR( aPListeJeuxY ,i);
               break;
         case 12:xrep = JEU_SEUIL_AVEC_RENONCEMENT(res1, i);
               break;
         case 13:xrep = MEFIANT(aPListeJeuxY);
               break;
         case 14:xrep = TIT_FOR_TAT_DUR(aPListeJeuxY, i);
               break;
```

```
    case 15:xrep = MAJ_CINQ(aPListeJeuxY, i);
         break;
    case 16:xrep = CONTINUE (aPListeJeuxY);
         break;
    case 17:xrep = PAVLOV (aPListeJeuxY, aPListeJeuxX);
         break;
    case 18:xrep = PAVLOV_CINQ (aPListeJeuxY, aPListeJeuxX,i);
         break;
    case 19:xrep = PROBA_MEME_QUE_LUI (aPListeJeuxY, aPListeJeuxX, i);
         break;
    case 20:xrep = PROBA_CONTINUE (aPListeJeuxY, aPListeJeuxX, i);
         break;
}

temp = (*Y).Strategie;
switch ( temp ){

    case 1: yrep = JEU_GENTILLE();
         break;
    case 2: yrep = JEU_MECHANTE();
         break;
    case 3: yrep = JEU_LUNATIQUE();
         break;
    case 4: yrep = JEU_TIT_FOR_TAT( aPListeJeuxX, i);
         break;
    case 5: yrep = JEU_RANCUNIERE(aPListeJeuxX, i);
         break;
    case 6: yrep = JEU_PERIODIQUE_MECHANTE (i);
         break;
    case 7: yrep = JEU_PERIODIQUE_GENTILLE (i);
         break;
    case 8: yrep = JEU_MAJ_MOU (aPListeJeuxX, i);
         break;
    case 9: yrep = JEU_SEUIL(res2, i);
         break;
    case 10:yrep = JEU_BINAIRE(i);
         break;
    case 11:yrep = JEU_MAJ_DUR( aPListeJeuxX ,i);
         break;
    case 12:yrep = JEU_SEUIL_AVEC_RENONCEMENT(res2, i);
         break;
    case 13:yrep = MEFIANT(aPListeJeuxX);
         break;
    case 14:yrep = TIT_FOR_TAT_DUR(aPListeJeuxX, i);
         break;
    case 15:yrep = MAJ_CINQ(aPListeJeuxX, i);
         break;
    case 16:yrep = CONTINUE (aPListeJeuxX);
         break;
    case 17:yrep = PAVLOV (aPListeJeuxX, aPListeJeuxY);
         break;
    case 18:yrep = PAVLOV_CINQ (aPListeJeuxX, aPListeJeuxY, i);
         break;
    case 19:yrep = PROBA_MEME_QUE_LUI (aPListeJeuxX, aPListeJeuxY, i);
         break;
    case 20:yrep = PROBA_CONTINUE (aPListeJeuxX, aPListeJeuxY, i);
         break;
}
```

```
    if ((yrep == 'r')||(xrep == 'r'))
    {
     res1 = res1 + rr;  //cc=3,tt=1, rr=2, tc=5,ct=0,
     res2 = res2 + rr;
    }
                else
    {
     if (yrep == 'c')
                 {
                        if (xrep == 'c')
                        {
                                res1 = res1 + cc;
                                res2 = res2 + cc;
                        }
                        else
                        {
                                res1 = res1 + tc;
                                res2 = res2 + ct;
                        }
                 }
                 else
                 {
                        if (xrep == 'c')
                        {
                                res1 = res1 + ct;
                                res2 = res2 + tc;
                        }
                        else
                        {
                                res1 = res1 + tt;
                                res2 = res2 + tt;
                        }
                 }
     }

    if (i==1)
    {
                aPListeJeuxX = new struct TJeu;  //si c'est la première fois qu'x et y jouent,
                (*aPListeJeuxX).Suivant = NULL;  //on spécifie qu'il n'y a pas de suivant car les jeux les
plus <<vieux>> seront en fin de liste.
                (*aPListeJeuxX).Coup=xrep;
                aPListeJeuxY = new struct TJeu;
                (*aPListeJeuxY).Suivant = NULL;
                (*aPListeJeuxY).Coup=yrep;
               }
    else
    {
     aPJeuX=new struct TJeu;   //on remplit le jeu joué par x et celui joué par y
                aPJeuY=new struct TJeu;
                (*aPJeuX).Coup=xrep;
                (*aPJeuY).Coup=yrep;
                (*aPJeuX).Suivant=aPListeJeuxX;   //si i=2 le pointeur vers le suivant du nouveau jeu de x
pointera vers le premier jeu de X
                (*aPJeuY).Suivant=aPListeJeuxY;
                aPListeJeuxX=aPJeuX;            //le premier jeu de la liste sera le dernier joué
                aPListeJeuxY=aPJeuY;
    }
    X=(*X).Suivant;
    Y=(*Y).Suivant;
```

```
        }
  SUPPRIMER_JEUX(aPListeJeuxX);
  SUPPRIMER_JEUX(aPListeJeuxY);

        *resultat1 = res1; //renvoie le résultat à l'adresse initialisée avant d'entrer dans la fonction
        *resultat2 = res2;
}

*********DEROULEMENT D'UN CONCOURS ************************

//Affrontements de chaque stratégie à tester face à des stratégies représentatives

CONCOURS () //le score de chaque stratégie testée est déterminé
{
 int i,c;
 int * aPResAffrStrategieTestee;
 int * aPResAffrStrategieRepresentative;

 aPResAffrStrategieTestee = new int;
 aPResAffrStrategieRepresentative = new int;
 for (i=1; i<NombreStrategiesTestees+1; i++)
  {
   if (NouvellesStrategies[i]==1)  // on teste si il s'agit d'une nouvelle stratégie (dont on n'a pas encore
déterminé le score)
   {
     ScoreStrategiesTestees[i]=0;
          for (c=1; c<NombreStrategiesRepresentatives+1; c++)
            {
                AFFRONTEMENT(ListeStrategiesTestees[i], ListeStrategiesRepresentatives[c],
aPResAffrStrategieTestee, aPResAffrStrategieRepresentative); //il faudra des pointeurs locaux qui ne
travaillent pas directement sur ceux pointant vers les gènes
                ScoreStrategiesTestees[i]=ScoreStrategiesTestees[i]+ ((* aPResAffrStrategieTestee) *
EffectifsStrategiesRepresentatives[c]);
                //on multiplie le score de la strategie testée face à cette représentative par le nombre de
rencontres quand des stratégies lunatiques se rencontrent ce n'est pas tout à fait bon entre ses effectifs et
ceux de la strategie représentative
                //ScoreStrategiesRepresentatives[c]=ScoreStrategiesRepresentatives[c]+ ((*
aPResAffrStrategieRepresentative)* EffectifsStrategiesTestees[i]* EffectifsStrategiesRepresentatives[c]) ;
     //on pourrait faire le même pour les stratégies représentatives si l'on désire faire évoluer leurs
effectifs
            }
     NouvellesStrategies[i]=0;
   }
  }
}


*********SELECTION DES STRATEGIES************

SELECTION()  //nous n'avons que le type de sélection en gardant des effectifs constants moins un effectif
à cause de l'arrondi
{
int i;
 long TotauxScoresTestees=0;
 double effreel, decimal;

 EffectifsTotauxTestees=0;
 for (i=1; i<NombreStrategiesTestees+1; i++) //on calcule le total des scores des stratégies testées
 {
        TotauxScoresTestees=TotauxScoresTestees+ScoreStrategiesTestees[i]*EffectifsStrategiesTestees[i];
```

```
        EffectifsTotauxTestees=EffectifsTotauxTestees+EffectifsStrategiesTestees[i];
    }

    for (i=1; i<NombreStrategiesTestees+1; i++)
    {
            if (TotauxScoresTestees != 0) //on calcule les effectifs de chaque stratégie en divisant son score par
le score d'un individu moyen; les effectifs d'une stratégie seront ainsi proportionnels au score qu'elle a
obtenu
        {
          effreel
=(EffectifsTotauxTestees*ScoreStrategiesTestees[i]*EffectifsStrategiesTestees[i])/TotauxScoresTestees;
//on fait les effectifs * le score des stratégies pour voir le score qu'elles obtiennent réellement, puisqu'après
l'affrontement on n'a le score que d'un individu de la stratégie
            decimal = modf(effreel, &EffectifsStrategiesTestees[i]); // les score de la testée prend la partie entière
        }
    }
}
```

**********DUPLIQUER LES CHROMOSOMES POUR GARDER LES CHROMOSOMES PARENTS APRES LE CROSSOVER******

```
DUPLIQUER(struct TGene * Strat, struct TGene * StratDup )
{
  int i;
  struct TGene * aPStrat;  //pointeur local
  struct TGene * aPStratDup, * bPStratDup;

  aPStrat=Strat;
  (* StratDup).Strategie=(* aPStrat).Strategie ;  //une fonction d'assignation d'un string
  aPStratDup=StratDup;

  for (i=2; i<NombreParts+1; i++) // pour i allant de 2 au nombre de parts jouées on rempli les
caractéristiques des gènes
  {
    aPStrat=(*aPStrat).Suivant;  //on avance sur le gène à duppliquer
            bPStratDup=new struct TGene; // on crée le gène qui suivra et ensuite on introduit ses
caractéristiques
            (* bPStratDup).Strategie=(* aPStrat).Strategie;
    (* aPStratDup).Suivant=bPStratDup; //lors du premier passage dans la boucle, le gène pointé par
StratDup et aPStratDup reçoit pour suivant le deuxième gène
     aPStratDup=(* aPStratDup).Suivant; //on avance sur le gène qui est en train de se duppliquer
  }
  (* aPStratDup).Suivant=NULL; //le dernier gène du chromosome duppliqué n'a pas de suivant
}
```

```
//*********SUPPRIMER LES STRATEGIES TESTEES ELIMINEES***********

SUPPRIMER( struct TGene * StratSup)     // fonction identique à SUPPRIMER_JEU mais conservée pour
plus de clareté.
{
  struct TGene * aPStratSup;
  int i;


  for (i=1; i<NombreParts+1; i++)
```

```
 {
  if (i<NombreParts)
    aPStratSup= (*StratSup).Suivant;
  delete StratSup;
  StratSup = aPStratSup;
 }
}
```

***********CROSSING_OVER PROPREMENT DIT***************

```
CROSSING_OVER()
{
 int i,c,Locus,co,Trouve,Endroit;
 struct TGene * aPStrategie1;
 struct TGene * aPStrategie2;
 struct TGene * bPStrategie1;
 struct TGene * bPStrategie2;
 struct TGene * cPStrategie1;


 for (i=1; i<NombreStrategiesApparaissants+1; i=i+2)
 {
        aPStrategie1=new struct TGene; //à la fin de la boucle ils repointent vers la valeur de départ avant
le new
  DUPLIQUER(ListeStrategiesTestees[StrategiesACroiser[i]],aPStrategie1); //!!!à un moment donné on
passait à DUPPLIQUER une adresse ne pointant pas vers un chromosome, quand il essaye de copier cela
se plantait évidemment
  aPStrategie2=new struct TGene;
  DUPLIQUER(ListeStrategiesTestees[StrategiesACroiser[i+1]],aPStrategie2);
        //on duplique les chromosomes avec lesquels on fait des crossing-over

  for (c=1; c<NombreCrossingOver+1; c++) //on réalise les crossing-over
  {
   bPStrategie1=aPStrategie1; //le pointeur a garde l'origine du chromosome sur lequel on travaille
   bPStrategie2=aPStrategie2;
   Locus = (random (NombreParts-1))+1; //on ne fait pas de crossing-over à partir du dernier gène
   for (co=1; co<Locus; co++)
   {
    bPStrategie1=(*bPStrategie1).Suivant; //on se positionne sur le gène à partir duquel on réalisera le
crossing-over
    bPStrategie2=(*bPStrategie2).Suivant;
   }
   cPStrategie1=(*bPStrategie1).Suivant;        //on réalise le crossing-over, exemple si le locus = 3 :
SSSYYYYY
   (*bPStrategie1).Suivant=(*bPStrategie2).Suivant;                                    // YYYSSSSS
                (*bPStrategie2).Suivant=cPStrategie1;
  }

  //ici on remplace des stratégies testées créées par d'autres nouvellement créées si les effectifs qu'elles
avaient obtenus précédemment se situent en dessous de EffectifLimite décidé par l'utilisateur
        c=NombreTesteesDepart+1;
        Trouve=0;
        while ((c<NombreStrategiesTestees+1)&&!(Trouve))
 {
  if (EffectifsStrategiesTestees[c]<EffectifLimite)
  {
                SUPPRIMER(ListeStrategiesTestees[c]);
```

```
                ListeStrategiesTestees[c]=aPStrategie1;
        NouvellesStrategies[c]=1;
            ScoreStrategiesTestees[c]=0;
                Trouve=1;
        Endroit=c;
     }
     c++;
    }
        if (!Trouve)
    {
                ListeStrategiesTestees[NombreStrategiesTestees+1]=aPStrategie1;  //remplacement des
stratégies ayant les plus petits effectifs par les nouvelles stratégies
      NouvellesStrategies[NombreStrategiesTestees+1]=1;
                ListeStrategiesTestees[NombreStrategiesTestees+2]=aPStrategie2;
      NouvellesStrategies[NombreStrategiesTestees+2]=1;
                NombreStrategiesTestees=NombreStrategiesTestees+2;
    }
    else
    {
          c=Endroit+1;
          while ((c<NombreStrategiesTestees+1)&&(Trouve==1))
     {
                if (EffectifsStrategiesTestees[c]<EffectifLimite)
      {
       SUPPRIMER(ListeStrategiesTestees[c]);
       ListeStrategiesTestees[c]=aPStrategie2;
       NouvellesStrategies[c]=1;
       ScoreStrategiesTestees[c]=0;
       Trouve=2;
      }
      c++;
     }
    }
        if (Trouve==1)
    {
                ListeStrategiesTestees[NombreStrategiesTestees+1]=aPStrategie2;
      NouvellesStrategies[NombreStrategiesTestees+1]=1;
                NombreStrategiesTestees=NombreStrategiesTestees+1;
    }
   }
}
```

*******SELECTION DES STRATEGIES SUBISSANT UN CROSS0VER ET DE CELLES QUI VONT DISPARAITRE****

```
CROISEMENT()
{
 int i,c,LaPremiere;
 int StrategieChoisie;
 int ScoreCumulesStrategiesTestees[10000],ScoreTotalTestees;

 //pour garder un même nombre de stratégies, nous en faisons disparaître autant que nous en croisons
 //avec les stratégies qui ont les plus grands effectifs qui ont plus de chance de se croiser et celles qui ont les
plus petits qui disparaissent

//choix aléatoire des stratégies à croiser

 for (i=1; i<NombreStrategiesTestees+1; i++)
```

```
        ScoreCumulesStrategiesTestees[i]=0;

 for (i=1; i<NombreStrategiesTestees+1; i++)
  {
   if (i==1)
    ScoreCumulesStrategiesTestees[1]=ScoreStrategiesTestees[1];
    else
          ScoreCumulesStrategiesTestees[i]=ScoreCumulesStrategiesTestees[i-1]+ScoreStrategiesTestees[i];
  }
  ScoreTotalTestees=ScoreCumulesStrategiesTestees[NombreStrategiesTestees];
 //remplissage du tableau des scores cumulés des stratégies testées; un nombre est tiré au hasard entre 0
et le score total des stratégies testées; la première stratégie dont le score cumulé est plus grand ou égal que
cette valeur est sélectionnée pour être croisée
for (i=1; i<(NombreStrategiesApparaissants+1); i++)
 {
        StrategieChoisie=random(ScoreTotalTestees)+1;
        c=1;
        while ((ScoreCumulesStrategiesTestees[c]<StrategieChoisie)&&(c<NombreStrategiesTestees))
//!!!cela se plantait parce que la condition était c<NombreStrategiesTestees+1 et donc quand c'était le
dernier élément qui était choisi, comme il augmente c de 1 ce sera en fait l'élément d'après qui sera choisi
et qui ne se trouve plus dans le tableau.
    c++;
   if (i % 2 == 1)
   {
    LaPremiere = c; //tous les nombres impairs, on a la première stratégie à croiser avec la deuxième, il
faut vérifier que cette dernière n'est pas la même pour éviter qu'une stratégie ne prenne le dessus sur les
autres
    StrategiesACroiser[i]=c;
   }
    else if ((i % 2 == 0)&&(c==LaPremiere))
     i--;
     else
      StrategiesACroiser[i]=c;
 }
 CROSSING_OVER();
}
```

**********AFFICHAGE DES MEILLEURES STRATEGIES*************

```
AFFICHER_MEILLEURE_STRATEGIE()

{
 int c,IndMax;
 int Max;
 struct TGene * StrategieAAfficher;


 Max=ScoreStrategiesTestees[1];
 IndMax=1;
 for (c=2; c<NombreStrategiesTestees+1; c++)
 {
  if (ScoreStrategiesTestees[c]>Max)
  {
   Max=ScoreStrategiesTestees[c];
   IndMax=c;
  }
 }
 StrategieAAfficher=ListeStrategiesTestees [IndMax];
```

```
 (*(*Form2).StringGrid2).Cells [1][NumTest]=ScoreStrategiesTestees[IndMax]/100;

 for (c=1; c<=NombreParts; c++)
  {
  (*(*Form2).StringGrid1).Cells [c][NumTest]=CORRESPONDANCE((*StrategieAAfficher).Strategie);
  //on compte les différents gènes de manière à obtenir leur total
  TotauxGenes[(*StrategieAAfficher).Strategie]=TotauxGenes[(*StrategieAAfficher).Strategie]+1;
       StrategieAAfficher =(*StrategieAAfficher).Suivant;
  }
}
```

***********DEVELOPPEMENT DES GENERATIONS*************

```
DEVELOPPEMENT_STRATEGIES()
{
 int i;

 for (i=1; i<NombreGenerations+1; i++)
 {
  //a l'intérieur de cette boucle il y a calcul des scores lors du concours et reproduction asexuée

       CONCOURS(); //détermine le score obtenu des stratégies en fonction de leurs effectifs
    SELECTION(); //suite au score obtenu, on détermine les nouveaux effectifs pour les différentes
stratégies
 }
}
```

//*******PROPORTION DES GENES AU SEIN DE LA STRATEGIE MOYENNE***********

```
PROPORTION_GENES()
{

  // remplissage de la form3
//calcul des proportions de gènes dans la stratégie moyenne
 (*(*Form3).StringGrid1).Cells[1][0]="Moyenne";
 (*(*Form3).StringGrid1).RowCount = (*(*Form1).StringGrid1).RowCount;
 for (int ligne =1; ligne<(*(*Form3).StringGrid1).RowCount ; ligne++)
 {
  (*(*Form3).StringGrid1).Cells [0][ligne]= (*(*Form1).StringGrid1).Cells [0][ligne];
  (*(*Form3).StringGrid1).Cells [1][ligne]=
FloatToStr(TotauxGenes[ligne]/StrToFloat((*(*Form1).NombreTests).Text)) ;
  (*(*(*Form3).Memo1).Lines).Add((*(*Form3).StringGrid1).Cells [1][ligne]);
 }
 (*(*Form3).NombreParties).Text=(*(*Form1).NombreParties).Text;
}


int CORRESPOND_INVERSE( AnsiString  nom)
{
    if (nom == "GENTIL")
     return 1;
     else if (nom== "MECHANT")
      return 2;
      else if (nom== "LUNATIQUE")
       return 3;
       else if (nom== "TIT_FOR_TAT")
        return 4;
```

```
        else if (nom== "RANCUNIER")
         return 5;
          else if (nom== "PERIOD_MECH")
          return 6;
           else if (nom== "PERIOD_GENT")
            return 7;
             else if (nom== "MAJ_MOU")
              return 8;
               else if (nom== "SEUIL")
                return 9;
                 else if (nom== "BINAIRE")
                  return 10;
                   else if (nom== "MAJ_DUR")
                    return 11;
                     else if (nom== "SEUILREN")
                      return 12;
                     else if (nom== "MEFIANT")
                      return 13;
                     else if (nom== "T_F_T_DUR")
                      return 14;
                     else if (nom== "MAJ_CINQ")
                      return 15;
                     else if (nom== "TESTCONTINUE")
                      return 16;
                     else if (nom== "PAVLOV")
                      return 17;
                     else if (nom== "PAV_CINQ")
                      return 18;
                     else if (nom== "PROB_MEME_QUE")
                      return 19;
                     else if (nom== "PROB_CONTINUER")
                      return 20;


}
```

***********PROPORTION DES GENES AU SEIN DE LA STRATEGIE MOYENNE***********

```
MEILLEUR_GENE_LOCAL()
{
 float Score[200];
 short int i, StrategieMajoritaire, colone, ligne;
 short int gene;
 AnsiString nom;

 //calcul du gène le mieux représenté à une position et de sa proportion à cette position
 (*(*Form3).StringGrid2).ColCount = (*(*Form2).StringGrid1).ColCount;
 for (colone =1; colone<=StrToInt((*(*Form1).NombreParties).Text); colone++)
  (*(*Form3).StringGrid2).Cells [colone][0]= "gène" + IntToStr(colone);
 (*(*Form3).StringGrid2).Cells [0][2]= "proportions :";

 for (colone =1; colone<=StrToInt((*(*Form1).NombreParties).Text); colone++)
 {
  for (i=1; i<=NombreStrategiesDifferentes; i++)
   Score[i]=0;
  for (ligne =1; ligne<=StrToInt((*(*Form1).NombreTests).Text); ligne++)
  {
   nom = (*(*Form2).StringGrid1).Cells[colone][ligne];
   gene =CORRESPOND_INVERSE(nom);
    Score [gene]=Score [gene] + 1;
  }
```

```
   StrategieMajoritaire=1;
   for (i = 2; i<=NombreStrategiesDifferentes; i++)
   {
    if (Score[StrategieMajoritaire]<Score[i])
      StrategieMajoritaire=i;
   }
   (*(*Form3).StringGrid2).Cells [colone][1]= CORRESPONDANCE (StrategieMajoritaire);
   (*(*Form3).StringGrid2).Cells [colone][2]= FloatToStr
(Score[StrategieMajoritaire]/StrToFloat((*(*Form1).NombreTests).Text));
   (*(*(*Form3).Memo2).Lines).Add((*(*Form3).StringGrid2).Cells [colone][1]);
  }
  for (colone =1; colone<=StrToInt((*(*Form1).NombreParties).Text); colone++)
   (*(*(*Form3).Memo2).Lines).Add((*(*Form3).StringGrid2).Cells [colone][2]);

}


//*********DEROULEMENT D'UN CONCOURS ECOLOGIQUE*****************

//(Affrontements de chaque stratégie face à des stratégies représentatives

CONCOURS_ECOLOGIQUE () //le score de chaque stratégie testée est déterminé
{
 int i,c;
 int * aPResAffrStrategieUne;
 int * aPResAffrStrategieDeux;

 aPResAffrStrategieUne = new int;
 aPResAffrStrategieDeux = new int;

 for (c=1; c<=NombreStrategiesTestees; c++)       //ici, NombreStrategiesTestees est le nombre de stratégies
représentatives +1
 {
   ScoreStrategiesTestees[c]=0;            //on met les scores des stratégies testées à 0
 }
 for (i=1; i<=NombreStrategiesTestees; i++)
  {
          for (c=i; c<=NombreStrategiesTestees; c++)
          {
                  AFFRONTEMENT(ListeStrategiesTestees[i], ListeStrategiesTestees[c],
aPResAffrStrategieUne, aPResAffrStrategieDeux);  //il faudra des pointeurs locaux qui ne travaillent pas
directement sur ceux pointant vers les gènes
          if (i==c)
          {
                  ScoreStrategiesTestees[i]=ScoreStrategiesTestees[i]+ ((* aPResAffrStrategieUne)*
(EffectifsStrategiesTestees[c]-1));
          } else
          {
                  ScoreStrategiesTestees[i]=ScoreStrategiesTestees[i]+ ((* aPResAffrStrategieUne)*
EffectifsStrategiesTestees[c]);
                  ScoreStrategiesTestees[c]=ScoreStrategiesTestees[c]+ ((* aPResAffrStrategieDeux)*
EffectifsStrategiesTestees[i]);
          }
    }
  }
}

int MEILLEURE_STRATEGIE_TESTS()
{
 int scoremax=0;
```

```
int meilleure=0;
for (int i=1; i<=StrToInt((*(*Form1).NombreTests).Text);i++)
{
  if ((*(*Form2).StringGrid2).Cells [1][i]>scoremax)
  {
    meilleure=i;
    scoremax=StrToInt((*(*Form2).StringGrid2).Cells [1][i]);
  }
}
return meilleure;
}

REMPLIR_LISTES_ECOLOGIQUES(int cas)
{
 short int i, c, compt=0;
 int meilleure;
 struct TGene * aLocPGene;
 struct TGene * bLocPGene;
 AnsiString nom;
 int gene;

 (*(*(*Form5).Memo1).Lines).Clear();
 NombreStrategiesTestees=0;
 for (i=1; i<=NombreStrategiesDifferentes; i++)
 {
  if (StrToInt((*(*Form1).StringGrid2).Cells[1][i])>0)
  {
    NombreStrategiesTestees++;
    compt++;

EffectifsStrategiesTestees[compt]=StrToInt((*(*Form1).StringGrid2).Cells[1][i])*EffectifsConstantsTestees/*E
ffectifsConstantsTestees*/;
    ListeStrategiesTestees [compt]=new struct TGene; // création du premier gène des stratégies à tester de
la liste remarquons que le tableau de pointeur est une variable globale pour jouer plus facilement avec les
stratégies
    REMPLIR_STRATEGIE(i, ListeStrategiesTestees [compt]); //on passe l'adresse de la structure en
paramètre variable et le numéro de ligne qui correspond à une stratégie
  }
 }
 NombreStrategiesTestees++;
 compt++;
 EffectifsStrategiesTestees[compt]=EffectifsConstantsTestees;
 ListeStrategiesTestees [compt]=new struct TGene;
 aLocPGene=ListeStrategiesTestees [compt];
 if (cas ==1)
 {
  meilleure = MEILLEURE_STRATEGIE_TESTS();
  Form5->Edit1->Text = "Test" + IntToStr (meilleure);
  nom = (*(*Form2).StringGrid1).Cells[1][meilleure];
  gene =CORRESPOND_INVERSE(nom);
  (*(*(*Form5).Memo1).Lines).Add(nom);
 }
 else
 {
  Form5->Edit1->Text = "Moyenne";
  nom = (*(*Form3).StringGrid2).Cells[1][1];
  gene =CORRESPOND_INVERSE(nom);
  (*(*(*Form5).Memo1).Lines).Add(nom);
 }
```

```
  (*aLocPGene).Strategie= gene;
  if (NombreParts==1)
    (*aLocPGene).Suivant=NULL;

  for (c=2; c<=NombreParts; c++)
  {
     if (cas ==1)
       nom = (*(*Form2).StringGrid1).Cells[c][meilleure];
     else
       nom = (*(*Form3).StringGrid2).Cells[c][1];
     gene =CORRESPOND_INVERSE(nom);
     (*(*(*Form5).Memo1).Lines).Add(nom);
     bLocPGene=new struct TGene; // on crée le gène qui suivra et ensuite on introduit ses caractéristiques
     (* bLocPGene).Strategie= gene; //si on veut affecter directement, il faut affecter caractère par caractère
cette fonction marche
     if (c==NombreParts)
       (* bLocPGene).Suivant= NULL; //on libère la mémoire vers laquelle pointe le pointeur du dernier
gène
     (* aLocPGene).Suivant=bLocPGene;
     aLocPGene=(* aLocPGene).Suivant;

  }

}

//*********MOYENNE DES SCORES DES MEILLEURES STRATEGIES**************

MOYENNE()
{
 int i, Moyenne, Somme=0;

 for (i=1; i<=(*(*Form2).StringGrid1).RowCount-1; i++)
   Somme=Somme+StrToInt((*(*Form2).StringGrid2).Cells [1][i]);
 Moyenne=Somme/(i-1);
 (*(*Form2).Edit1).Text=IntToStr(Moyenne);
}
```

Module coordinateur (algorithmes génétiques)

```
//---------------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop

#include "Strate1FP.h"
#include "Strate2FP.h"
#include "Strate3FP.h"
#include "Strate4FP.h"
#include "Strate6FP.h"
#include "Strate7FP.h"
#include "StrateFP.h"

//---------------------------------------------------------------------------
#pragma link "Grids"
#pragma resource "*.dfm"

# include <iostream.h>
# include <string.h>
```

```
# include <stdlib.h>
# include <conio.h>
# include <stdio.h>



TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
  (*StringGrid1).RowCount= NombreStrategiesDifferentes+1;
  (*StringGrid2).RowCount= NombreStrategiesDifferentes+1;
  // remplissage de la grille des stratégies à tester
 for (int c=1; c<=NombreStrategiesDifferentes; c++)
 {
    (*StringGrid1).Cells[0][c]=CORRESPONDANCE(c);
    (*StringGrid1).Cells[1][c]=1;
 }
  //remplissage de la grille des stratégies représentatives
 for (int c=1; c<=NombreStrategiesDifferentes; c++)
 {
    (*StringGrid2).Cells[0][c]=CORRESPONDANCE(c);
    (*StringGrid2).Cells[1][c]=0;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
 void close();
}


//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 int reussit=1;

 //création des colones et lignes de la grille1 de la form2 et des libellés des colones et lignes fixes
 (*(*Form2).StringGrid1).ColCount = StrToInt((*NombreParties).Text)+1;
 (*(*Form2).StringGrid1).RowCount = StrToInt((*NombreTests).Text)+1;
 for (int ligne =1; ligne<=StrToInt((*NombreTests).Text); ligne++)
  (*(*Form2).StringGrid1).Cells [0][ligne]= "test" + IntToStr(ligne);
 for (int colone =1; colone<=StrToInt((*NombreParties).Text); colone++)
  (*(*Form2).StringGrid1).Cells [colone][0]= "gène" + IntToStr(colone);
 //création des lignes et colones de la grille2 de la form2
 (*(*Form2).StringGrid2).ColCount = 2;
 (*(*Form2).StringGrid2).RowCount = StrToInt((*NombreTests).Text)+1;
 for (int ligne =1; ligne<=StrToInt((*NombreTests).Text); ligne++)
  (*(*Form2).StringGrid2).Cells [0][ligne]= "test" + IntToStr(ligne);
 for (int ligne =1; ligne<=StrToInt((*NombreTests).Text); ligne++)
  (*(*Form2).StringGrid2).Cells [0][ligne]= "test" + IntToStr(ligne);
 (*(*Form2).StringGrid2).Cells [1][0]= "score";

*********PROGRAMME PRINCIPAL*********


 randomize(); // initialisation du générateur aléatoire
 NombreParts=StrToInt((*NombreParties).Text);
 NombreGenerations=1;
```

```
NombreCroisements=StrToInt((*NombreCrois).Text);
NombreCrossingOver=StrToInt((*NombreCrossOver).Text);
NombreStrategiesApparaissants=2*StrToInt((*NombreStrategiesApps).Text);
(*(*(*Form3).Memo1).Lines).Clear();
(*(*(*Form3).Memo2).Lines).Clear();

for (c=1; c<=1000; c++)
  NouvellesStrategies[c]=0;
for (c=1; c<=NombreStrategiesDifferentes; c++)
  TotauxGenes[c]=0;
(*ProgressBar1).Max = StrToInt((*NombreTests).Text)*(NombreCroisements+1)+1;
for (NumTest=1; NumTest<=StrToInt((*NombreTests).Text); NumTest++)
{
 REMPLIR_LISTES();
 if (NombreStrategiesTestees>0)
 {
  if (! ((NombreStrategiesTestees==1) && (NombreCroisements>0)))
  {
   reussit =1;
   for (c=1; c<=NombreStrategiesTestees; c++)
     NouvellesStrategies[c]=1;
   for (i=1; i<NombreCroisements+2; i++)  //on développe une fois en plus les stratégies qu'il n'y a de
croisement pour voir le score des nouvelles stratégies créées ou voir le score de stratégies sans pour autant
faire un croisement
    {
      DEVELOPPEMENT_STRATEGIES();
      ProgressBar1->Position = ProgressBar1->Position+1;
      if (i<NombreCroisements+1)
      {
        CROISEMENT();
        for (c=1; c<NombreStrategiesTestees+1; c++)
          EffectifsStrategiesTestees[c]=EffectifsConstantsTestees;
      }
    }
    AFFICHER_MEILLEURE_STRATEGIE();
    for (i=1; i<=NombreStrategiesTestees; i++)
      SUPPRIMER(ListeStrategiesTestees[i]);
    for (i=1; i<=NombreStrategiesRepresentatives; i++)
      SUPPRIMER(ListeStrategiesRepresentatives[i]);
    ProgressBar1->Position = ProgressBar1->Position+1;
   }
   else
     { reussit = 0;
      Form7->Show();
     }
 }
 else
    { reussit = 0;
     Form6->Show();
    }
}
if (reussit)
{
  MOYENNE();
  (*Form3).Show();
  ProgressBar1->Position = 0;


  PROPORTION_GENES();
  MEILLEUR_GENE_LOCAL();
```

```
  }
}

//---------------------------------------------------------------------------
void __fastcall TForm1::GroupBox2Click(TObject *Sender)
{
// (*StringGrid1).Cells[0][1]="gentille";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------------
```

Module coordinateur (confrontations écologiques)

```
//---------------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop

#include "Strate5FP.h"
#include "Strate3FP.h"
#include "Strate2FP.h"
#include "Strate1FP.h"
#include "StrateFP.h"

//---------------------------------------------------------------------------
#pragma link "Grids"
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn2Click(TObject *Sender)
{
Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn3Click(TObject *Sender)
{
 int numstrat;
 NombreParts=StrToInt((*NombreParties).Text);
 NombreGenerations=500;

 (*ProgressBar1).Max = NombreGenerations+1;
 ProgressBar1->Position = 1;
 REMPLIR_LISTES_ECOLOGIQUES(1);

 for (int c=1; c< NombreStrategiesTestees; c++)
  {
```

```
    numstrat = (*ListeStrategiesTestees [c]).Strategie;
    (*(*Form5).StringGrid1).Cells[0][c] = CORRESPONDANCE(numstrat) ;
  }
 (*(*Form5).StringGrid1).Cells[0][NombreStrategiesTestees]= "ALG_GEN";
 (*(*Form5).StringGrid1).RowCount= NombreStrategiesTestees+1;
 (*(*Form5).StringGrid1).ColCount= NombreGenerations+1;

 for (int i=1; i<=NombreGenerations; i++)
 {

  CONCOURS_ECOLOGIQUE();
  SELECTION();
  for (int c=1; c<= NombreStrategiesTestees; c++)
  {
    (*(*Form5).StringGrid1).Cells [i][c] = EffectifsStrategiesTestees[c];
  }
  (*(*Form5).StringGrid1).Cells [i][0] = "T"+IntToStr(i) ;
  ProgressBar1->Position = i+1;

 }
 for (i=1; i<=NombreStrategiesTestees; i++)
   SUPPRIMER(ListeStrategiesTestees[i]);
 Form5->Show();
 ProgressBar1->Position = 0;

}


//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn4Click(TObject *Sender)
{
 int numstrat;
 NombreParts=StrToInt((*NombreParties).Text);
 NombreGenerations=500;

 (*ProgressBar1).Max = NombreGenerations+1;
 ProgressBar1->Position = 1;
 REMPLIR_LISTES_ECOLOGIQUES(2);

 for (int c=1; c< NombreStrategiesTestees; c++)
  {
    numstrat = (*ListeStrategiesTestees [c]).Strategie;
    (*(*Form5).StringGrid1).Cells[0][c] = CORRESPONDANCE(numstrat) ;
  }
 (*(*Form5).StringGrid1).Cells[0][NombreStrategiesTestees]= "ALG_GEN";
 (*(*Form5).StringGrid1).RowCount= NombreStrategiesTestees+1;
 (*(*Form5).StringGrid1).ColCount= NombreGenerations+1;

 for (int i=1; i<=NombreGenerations; i++)
 {

  CONCOURS_ECOLOGIQUE();
  SELECTION();
  for (int c=1; c<= NombreStrategiesTestees; c++)
  {
    (*(*Form5).StringGrid1).Cells [i][c] = EffectifsStrategiesTestees[c];
  }
  (*(*Form5).StringGrid1).Cells [i][0] = "T"+IntToStr(i) ;
  ProgressBar1->Position = i+1;
 }
```

```
  for (i=1; i<=NombreStrategiesTestees; i++)
     SUPPRIMER(ListeStrategiesTestees[i]);
  Form5->Show();
  ProgressBar1->Position = 0;

}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
Form3->Hide();
}
//---------------------------------------------------------------------------
```

Fenêtre « choix d'un environnement type »

```
//---------------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop

#include "Strate1FP.h"
#include "Strate4FP.h"
#include "Strate5FP.h"

//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::BitBtn1Click(TObject *Sender)
{
int x;

   x=ListBox1->ItemIndex;
   switch (x){
   case 0:
      {for (int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
       (*(*Form1).StringGrid2).Cells[1][i]=1;
       Form5->Edit2->Text= "toutes ";}
      break;

   case 1:
      {for (int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
       (*(*Form1).StringGrid2).Cells[1][i]=0;
       (*(*Form1).StringGrid2).Cells[1][2]=1;
       (*(*Form1).StringGrid2).Cells[1][3]=1;
       (*(*Form1).StringGrid2).Cells[1][6]=1;
       (*(*Form1).StringGrid2).Cells[1][7]=1;
       (*(*Form1).StringGrid2).Cells[1][10]=1;
       (*(*Form1).StringGrid2).Cells[1][11]=1;
       (*(*Form1).StringGrid2).Cells[1][13]=1;
       Form5->Edit2->Text= "pouvant prendre l'initiative de trahir ";
      }
```

```
    break;

case 2:
    {for (int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=1;
     (*(*Form1).StringGrid2).Cells[1][2]=0;
     (*(*Form1).StringGrid2).Cells[1][3]=0;
     (*(*Form1).StringGrid2).Cells[1][6]=0;
     (*(*Form1).StringGrid2).Cells[1][7]=0;
     (*(*Form1).StringGrid2).Cells[1][10]=0;
     (*(*Form1).StringGrid2).Cells[1][11]=0;
     (*(*Form1).StringGrid2).Cells[1][13]=0;
     Form5->Edit2->Text= "ne prenant jamais l'initiative de trahir ";
     }
    break;
case 3:
    {for (int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=0;
     (*(*Form1).StringGrid2).Cells[1][1]=1;
     (*(*Form1).StringGrid2).Cells[1][2]=1;
     (*(*Form1).StringGrid2).Cells[1][3]=1;
     (*(*Form1).StringGrid2).Cells[1][6]=1;
     (*(*Form1).StringGrid2).Cells[1][7]=1;
     (*(*Form1).StringGrid2).Cells[1][10]=1;
     Form5->Edit2->Text= "à jeux indépendants de celui de l'adversaire ";
     }
    break;
case 4:
    {for (int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=1;
     (*(*Form1).StringGrid2).Cells[1][1]=0;
     (*(*Form1).StringGrid2).Cells[1][2]=0;
     (*(*Form1).StringGrid2).Cells[1][3]=0;
     (*(*Form1).StringGrid2).Cells[1][6]=0;
     (*(*Form1).StringGrid2).Cells[1][7]=0;
     (*(*Form1).StringGrid2).Cells[1][10]=0;
     Form5->Edit2->Text= "à jeux dépendants de celui de l'adversaire ";
    }
    break;
case 5:
    {for(int i=1; i<=(*(*Form1).StringGrid2).RowCount;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=0;
     (*(*Form1).StringGrid2).Cells[1][17]=1;
     (*(*Form1).StringGrid2).Cells[1][18]=1;
     (*(*Form1).StringGrid2).Cells[1][19]=1;
     (*(*Form1).StringGrid2).Cells[1][20]=1;
     Form5->Edit2->Text= "à jeux dépendants de la réponse de l'adversaire à leurs jeux ";
    }
    break;
case 6:
    {for(int i=1; i<=14;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=1;
    (*(*Form1).StringGrid2).Cells[1][9]=0;
    (*(*Form1).StringGrid2).Cells[1][10]=0;
    (*(*Form1).StringGrid2).Cells[1][12]=0;
    for(int i=15; i<=(*(*Form1).StringGrid2).RowCount;i++)
     (*(*Form1).StringGrid2).Cells[1][i]=0;
    Form5->Edit2->Text= "de Delahaye et Mathieu ";
    }
    break;
```

```
    }
    Form4->Hide();

}
//---------------------------------------------------------------------------
void __fastcall TForm4::BitBtn2Click(TObject *Sender)
{
Form4->Hide();
}
//---------------------------------------------------------------------------
```